NITHISHKUMAR(192124111)

DSA0202-COMPUTER VISION WITH OPENCV

IMAGE HANDLING BASICS

Aim:

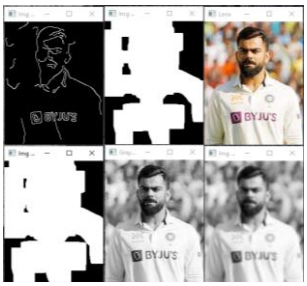
Write a python program to Handle an image using opencv

Program:

```
import cv2
import numpy as np
kernel = np.ones((5,5),np.uint8)
print(kernel)
path = "E:/1.jpg"
img = cv2.imread(path)
imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
imgBlur = cv2.GaussianBlur(imgGray,(7,7),0)
imgCanny = cv2.Canny(imgBlur,100,200)
imgDilation = cv2.dilate(imgCanny,kernel , iterations = 10)
imgEroded = cv2.erode(imgDilation,kernel,iterations=2)
cv2.imshow("Lena",img)
cv2.imshow("GrayScale",imgGray)
cv2.imshow("Img Blur",imgBlur)
cv2.imshow("Img Canny",imgCanny)
cv2.imshow("Img Dialation",imgDilation)
cv2.imshow("Img Erosion",imgEroded)
cv2.waitKey(0)
```

Input:

output:



1.Grayscale

Aim:

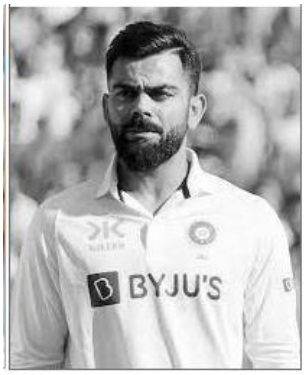
Write a python program to grayscale an image using opencv

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imshow("Original", img)
cv2.imshow("Grayscale", gray)
cv2.waitKey(0)
```

input:

output:



2.Gaussian Blur image

Aim:

Write a python program to blur an image using opencv

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
blur = cv2.GaussianBlur(img, (5, 5), 0)
cv2.imshow("Original", img)
cv2.imshow("Blurred", blur)
cv2.waitKey(0)
```

input:

output:

3.Canny edge

Aim:

Write a python program to detect Edge In an image using opencv

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
edges = cv2.Canny(img, 100, 200)
cv2.imshow("Original", img)
cv2.imshow("Edges", edges)
cv2.waitKey(0)
```

input:

output:

4.Dilated

Aim:

Write a python program to Dilate an image using opencv

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
edges = cv2.Canny(img, 100, 200)
kernel = np.ones((5, 5), np.uint8)
dilated = cv2.dilate(img, kernel, iterations=1)
cv2.imshow("Original", img)
cv2.imshow("Dilated", dilated)
cv2.waitKey(0)
```

input:

output:

5.Eroded

Aim:

Write a python program to erode an image using opencv

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
kernel = np.ones((5, 5), np.uint8)
eroded = cv2.erode(img, kernel, iterations=1)
cv2.imshow("Original", img)
cv2.imshow("Eroded", eroded)
cv2.waitKey(0)
```

input:

output:

6.video captured

Aim:

To Read captured video in python and display the video, in slow motion and in fast motion.

Program:

```
import cv2

cap = cv2.VideoCapture(0)

if not cap.isOpened():
    print("Could not open video capture device")
    exit()
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

cv2.namedWindow("Video", cv2.WINDOW_NORMAL)
cv2.resizeWindow("Video", width, height)

delay = 0
speed_factor = 1

while True:
    ret, frame = cap.read()
    if not ret:
        print("Could not read frame from video capture device")
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    edges = cv2.Canny(gray, 50, 150)

    cv2.imshow("Video", frame)

    if speed_factor > 1:
        cv2.waitKey(delay * speed_factor)
    elif speed_factor < 1:
        cv2.waitKey(int(delay / speed_factor))
    else:
        cv2.waitKey(delay)
    key = cv2.waitKey(1)
    if key == ord('q'):
        break
    elif key == ord('f'):
        speed_factor = min(speed_factor * 2, 8)
    elif key == ord('s'):

        speed_factor = max(speed_factor / 2, 0.25)

cap.release()
cv2.destroyAllWindows()
```

Result:

captured video in python and display the video, in slow motion and in fast motion is successfully executed

7.web capture

Aim:

To **Capture video from web Camera and Display the video, in slow motion and in fast motion**

Program:

```
import cv2
cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Could not open video capture device")
    exit()

width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
cv2.namedWindow("Video", cv2.WINDOW_NORMAL)
cv2.resizeWindow("Video", width, height)
while True:
    ret, frame = cap.read()
    if not ret:
        print("Could not read frame from video capture device")
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    cv2.rectangle(frame, (100, 100), (200, 200), (0, 0, 255), 2)
    cv2.imshow("Video", frame)
    cv2.imshow("Grayscale", gray)
    key = cv2.waitKey(1)
    if key == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Result:

captured video in python and display the video, in slow motion and in fast motion is successfully executed

8. smaller bigger

Aim:

Write a python program to scale an image using opencv

Program:

```
import cv2

img = cv2.imread("E:/1.jpg")

height, width = img.shape[:2]

scale_factor = 1.5

bigger_img = cv2.resize(img, (int(width * scale_factor), int(height * scale_factor)))

scale_factor = 0.5

smaller_img = cv2.resize(img, (int(width * scale_factor), int(height *
scale_factor)))

cv2.imshow("Original Image", img)

cv2.imshow("Bigger Image", bigger_img)

cv2.imshow("Smaller Image", smaller_img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

input:

output:

9.Rotation

Aim:

Write a python program to rotate an image using opencv

Program:

```
import cv2
img = cv2.imread("E:/1.jpg")
height, width = img.shape[:2]
angle = 30
center = (width/2, height/2)
M = cv2.getRotationMatrix2D(center, angle, 1)
clockwise_img = cv2.warpAffine(img, M, (width, height))
M = cv2.getRotationMatrix2D(center, -angle, 1)
counter_clockwise_img = cv2.warpAffine(img, M, (width, height))
cv2.imshow("Original Image", img)
cv2.imshow("Clockwise Rotated Image", clockwise_img)
cv2.imshow("Counter-Clockwise Rotated Image", counter_clockwise_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

input:

output:

10.image move

Aim:

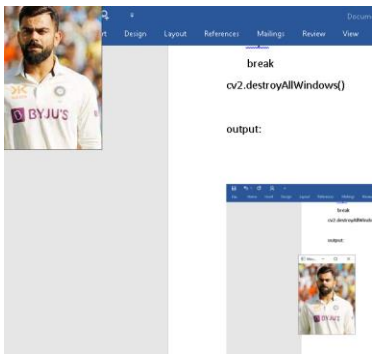
Write a python program to move an image using opencv

Program:

```
import cv2
img = cv2.imread("E:/1.jpg")
x, y = 100, 100
dx, dy = 50, 50
def move_image():
    global x, y, dx, dy
    x += dx
    y += dy
    if x < 0 or x > img.shape[1] or y < 0 or y > img.shape[0]:
        dx = -dx
        dy = -dy
def draw_image():
    global x, y
    cv2.imshow("Moving Image", img)
    cv2.moveWindow("Moving Image", x, y)
draw_image()
while True:
    move_image()
    draw_image()
    key = cv2.waitKey(50)
    if key != -1:
        break
cv2.destroyAllWindows()
```

Input:

output:



11.Affine Transformation

Aim:

Write a python program to Perform Affine Transformation on the image

Program:

```
import cv2
import numpy as np

img = cv2.imread("1.jpg")
M = np.float32([[0.5, 0.5, 50], [0.5, -0.5, 50]])
dst = cv2.warpAffine(img, M, (img.shape[1], img.shape[0]))
cv2.imshow("Original Image", img)
cv2.imshow("Transformed Image", dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

12.prespective Transformation

Aim:

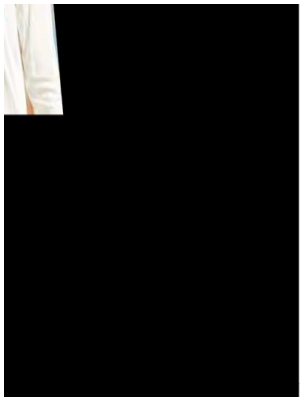
Write a python program to Perform Perspective Transformation on the image

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
roi_points = np.array([(150, 200), (450, 200), (550, 500), (50, 500)])
target_points = np.array([(0, 0), (400, 0), (400, 600), (0, 600)])
M = cv2.getPerspectiveTransform(roi_points.astype(np.float32), target_points.ast
dst = cv2.warpPerspective(img, M, (400, 600))
cv2.imshow("Original Image", img)
cv2.imshow("Transformed Image", dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:



13.prespective Transformation on video

Aim:

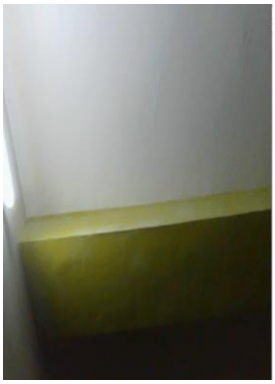
Write a python program to Perform Perspective Transformation on the image

Program:

```
import cv2
import numpy as np
roi_points = np.array([(150, 200), (450, 200), (550, 500), (50, 500)])
target_points = np.array([(0, 0), (400, 0), (400, 600), (0, 600)])
M = cv2.getPerspectiveTransform(roi_points.astype(np.float32), target_points.ast
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    if not ret:
        break
    dst = cv2.warpPerspective(frame, M, (400, 600))
    cv2.imshow("Original Frame", frame)
    cv2.imshow("Transformed Frame", dst)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Input:

output:



14.Homography matrix

Aim:

Write a python program to Perform transformation using Homography matrix.

Program:

```
import cv2
import numpy as np
image = cv2.imread('F:/2.jpg')
target_points = np.array([[0, 0], [500, 0], [500, 500], [0, 500]], dtype=np.floa
source_points = np.array([[141, 131], [480, 159], [493, 630], [64, 601]], dtype=
homography_matrix, _ = cv2.findHomography(source_points, target_points)
transformed_image = cv2.warpPerspective(image, homography_matrix, (500, 500))
cv2.imshow('Original Image', image)
cv2.imshow('Transformed Image', transformed_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

15.Direct Linear Transformation

Aim:

Write a python program to Perform transformation using Direct Linear Transformation

Program:

```
import cv2
import numpy as np

image = cv2.imread('E:/1.jpg')
source_points = np.array([[141, 131], [480, 159], [493, 630], [64, 601]], dtype=np.float32)
target_points = np.array([[0, 0], [500, 0], [500, 500], [0, 500]], dtype=np.float32)

num_points = source_points.shape[0]
A = np.zeros((2*num_points, 9), dtype=np.float64)
for i in range(num_points):
    x, y = source_points[i]
    u, v = target_points[i]
    A[2*i] = [x, y, 1, 0, 0, 0, -u*x, -u*y, -u]
    A[2*i+1] = [0, 0, 0, x, y, 1, -v*x, -v*y, -v]

_, _, V = np.linalg.svd(A)
h = V[-1, :]
homography_matrix = h.reshape((3, 3))

transformed_image = cv2.warpPerspective(image, homography_matrix, (500, 500))

cv2.imshow('Original Image', image)
cv2.imshow('Transformed Image', transformed_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

16.Edge detection using Canny Method

Aim:

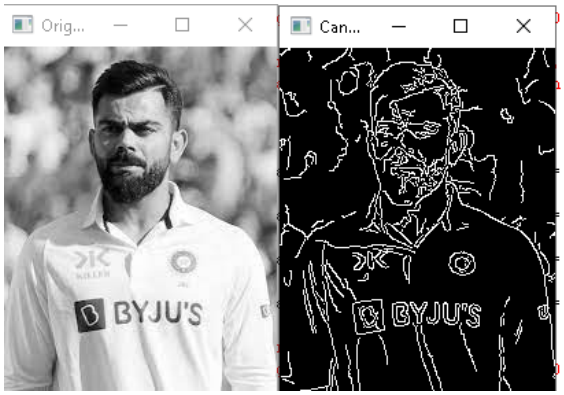
Write a python program to Perform Edge detection using canny method

Program:

```
import cv2
import numpy as np
img = cv2.imread('E:/1.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (3, 3), 0)
edges = cv2.Canny(blur, 100, 200)
cv2.imshow('Canny Edge Detection', edges)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output



17. Edge detection using Sobel Matrix along X axis

Aim:

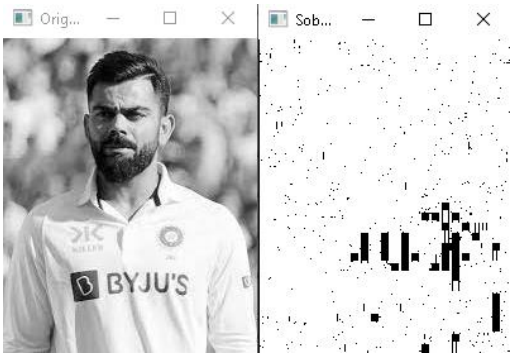
Write a python program to Perform Edge detection using Sobel Matrix along X axis

Program:

```
#17.Edge detection using Sobel Matrix along X axis
import cv2
import numpy as np
img = cv2.imread('E:/1.jpg', 0)
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
sobelx = np.abs(sobelx)
cv2.imshow('Original Image', img)
cv2.imshow('Sobel Edge Detection (X-axis)', sobelx)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:



18. Edge detection using Sobel Matrix along Y axis

Aim:

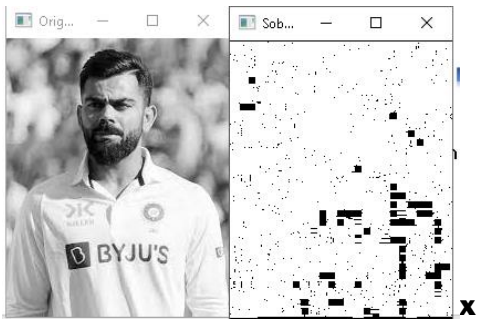
Write a python program to Perform Edge detection using Sobel Matrix along Y axis

Program:

```
import cv2
import numpy as np
img = cv2.imread('E:/1.jpg', 0)
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
sobely = np.abs(sobely)
cv2.imshow('Original Image', img)
cv2.imshow('Sobel Edge Detection (Y-axis)', sobely)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:



19. Edge detection using Sobel Matrix along XY axis

Aim:

Write a python program to Edge detection using Sobel Matrix along XY axis a image using opencv

Program:

```
import cv2
import numpy as np
img = cv2.imread('E:/1.jpg', cv2.IMREAD_GRAYSCALE)
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
sobel_combined = cv2.addWeighted(sobelx, 0.5, sobely, 0.5, 0)
cv2.imshow('Sobel Edge Detection', sobel_combined)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

20. Sharpening of Image using Laplacian mask with negative a center coefficient.

Aim:

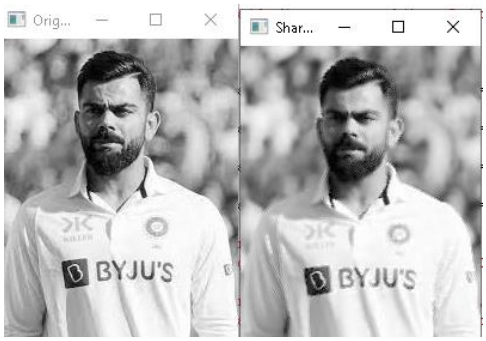
Write a python program to Perform Sharpening of Image using Laplacian mask with negative center coefficient

Program:s

```
import cv2
import numpy as np
img = cv2.imread('E:/1.jpg', cv2.IMREAD_GRAYSCALE)
laplacian_filter = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
sharpened = cv2.filter2D(img, -1, laplacian_filter)
sharpened_img = cv2.addWeighted(img, 1.5, sharpened, -0.5, 0)
cv2.imshow('Original Image', img)
cv2.imshow('Sharpened Image', sharpened_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:



21. Sharpening of Image using Laplacian mask with extension of neighnour diagonals

Aim:

Write a python program to Sharpening of Image using Laplacian mask with extension of neighbor diagonals .

Program:

```
import cv2
import numpy as np
image = cv2.imread('E:/1.jpg')
kernel = np.array([[0, -1, 0],
                   [-1, 8, -1],
                   [0, -1, 0]])
sharpened = cv2.filter2D(image, -1, kernel)
cv2.imshow('Original', image)
cv2.imshow('Sharpened', sharpened)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

22. Laplacian mask with positive center coefficient

Aim:

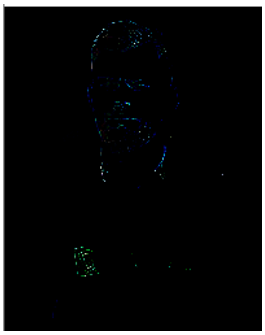
Write a python program to perform laplacian mask with positive center coefficient

Program:

```
import cv2
import numpy as np
image = cv2.imread('E:/1.jpg')
kernel = np.array([[0, 1, 0],
                   [1, -8, 1],
                   [0, 1, 0]])
sharpened = cv2.filter2D(image, -1, kernel)
cv2.imshow('Original', image)
cv2.imshow('Sharpened', sharpened)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:



23. Sharpening of Image using unsharp masking.

Aim:

Write a python program to Perform Sharpening of Image using unsharp masking.

Program:

```
import cv2
import numpy as np
image = cv2.imread('F:/3.jpg')
kernel = np.array([[0, 1, 0],
                   [1, -4, 1],
                   [0, 1, 0]])
sharpened = cv2.filter2D(image, -1, kernel)
cv2.imshow('Original', image)
cv2.imshow('Sharpened', sharpened)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

24.High boost mask

Aim:

Write a python program to Perform Sharpening of Image using High-Boost Masks

Program:

```
import cv2
import numpy as np
img = cv2.imread('E:/1.jpg', cv2.IMREAD_GRAYSCALE)
blur = cv2.GaussianBlur(img, (5,5), 0)
high_pass = img - blur
A = 2.0
high_boost = A * high_pass
sharpened = img + high_boost
cv2.imshow('Original', img)
cv2.imshow('Sharpened', sharpened)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

25.image gradient using masking

Aim:

Write a python program to Perform Sharpening of Image using Gradient masking

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
kernel = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
sharpened_img = cv2.filter2D(img, -1, kernel)
cv2.imshow('Input Image', img)
cv2.imshow('Sharpened Image', sharpened_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

26.insert water mark

Aim:

Write a python program to Insert water marking to the image using OpenCV

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
watermark = cv2.imread("watermark.png")
watermark = cv2.resize(watermark, (img.shape[1], img.shape[0]))
watermarked = cv2.addWeighted(img, 1, watermark, 0.5, 0)
cv2.imwrite('watermarked_image.png', watermarked)
```

Input:

output:

27.copying and pasting a image inside another image

Aim:

Write a python program to Copying and pasting image inside another image using OpenCV.

Program:

```
import cv2
import numpy as np
img1 = cv2.imread("E:/1.jpg")
img2 = cv2.imread("F:/2.jpg")
x, y, w, h = 100, 100, 200, 200
roi = img1[y:y+h, x:x+w]
roi_resized = cv2.resize(roi, (w, h))
img2gray = cv2.cvtColor(roi_resized,cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img2gray, 10, 255, cv2.THRESH_BINARY)
mask_inv = cv2.bitwise_not(mask)
img2_bg = cv2.bitwise_and(img2,img2,mask = mask_inv)
img1_fg = cv2.bitwise_and(roi_resized,roi_resized,mask = mask)
dst = cv2.add(img2_bg,img1_fg)
img2[y:y+h, x:x+w] = dst
cv2.imshow('Result', img2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

28.convolution kernel

Aim:

Write a python program to . Find the boundary of the image using Convolution kernel for the given image.

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
kernel = np.array([[-1,-1,-1],[-1,9,-1],[-1,-1,-1]])
sharpened_img = cv2.filter2D(img, -1, kernel)
cv2.imshow('Input Image', img)
cv2.imshow('Sharpened Image', sharpened_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

input:

output:

29.Erosion technique

Aim:

Write a python program to erosion technique a image using opencv

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg", cv2.IMREAD_GRAYSCALE)
kernel = np.ones((5, 5), np.uint8)
eroded_img = cv2.erode(img, kernel, iterations=1)
cv2.imshow("Original Image", img)
cv2.imshow("Eroded Image", eroded_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

30.dilation technique

Aim:

Write a python program to Morphological operations based on OpenCV using Dilation technique.

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
kernel = np.ones((5,5),np.uint8)
dilation = cv2.dilate(img,kernel,iterations = 1)
cv2.imshow('Original', img)
cv2.imshow('Dilated', dilation)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

31.opening technique

Aim:

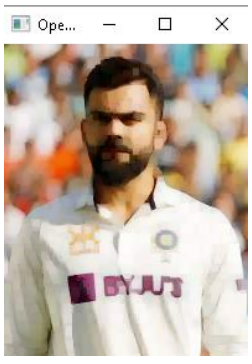
Write a python program to apply Morphological operations based on OpenCV using Opening technique.

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg")
kernel = np.ones((5,5),np.uint8)
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
cv2.imshow('Original', img)
cv2.imshow('Opened', opening)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:



32.closing technique

Aim:

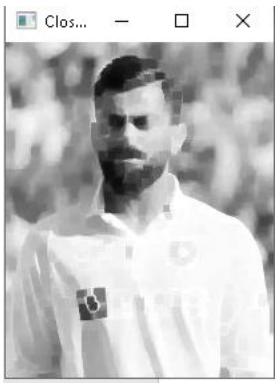
Write a python program to apply Morphological operations based on OpenCV using closing technique.

Program:

```
import cv2
import numpy as np
img = cv2.imread("E:/1.jpg", cv2.IMREAD_GRAYSCALE)
kernel = np.ones((5,5), np.uint8)
closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)
cv2.imshow('Original', img)
cv2.imshow('Closing', closing)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:



33.morphological gradient

Aim:

Write a python program to apply morphological gradient in an image using opencv

Program:

```
import cv2
import numpy as np
img = cv2.imread('E:/1.jpg', 0)
kernel = np.ones((3,3), np.uint8)
gradient = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)
cv2.imshow('Morphological Gradient', gradient)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

34.Top hat

Aim:

Write a python program to top hat a image using opencv

Program:

```
import cv2
import numpy as np
img = cv2.imread('E/1.jpg', cv2.IMREAD_GRAYSCALE)
kernel = np.ones((5,5), np.uint8)
top_hat = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel)
cv2.imshow('Output', top_hat)
cv2.waitKey(0)
```

Input:

output:

35.Black hat

Aim:

Write a python program to Black hat an image using opencv

Program:

```
import cv2
import numpy as np

img = cv2.imread('E/1.jpg', cv2.IMREAD_GRAYSCALE)
kernel = np.ones((5,5), np.uint8)
black_hat = cv2.morphologyEx(img, cv2.MORPH_BLACKHAT, kernel)
cv2.imshow('Output', black_hat)
cv2.waitKey(0)
```

Input:

output:

36. Object recognition

Aim:

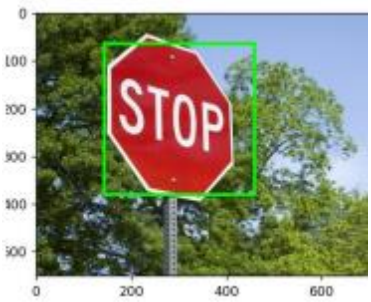
To Recognise object from the given image by general Object recognition using OpenCV.

Program:

```
#Object Recognition
import cv2
from matplotlib import pyplot as plt

img = cv2.imread("3.jpg")
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
stop_data = cv2.CascadeClassifier('stop_data.xml')

found = stop_data.detectMultiScale(img_gray,minSize =(20, 20))

amount_found = len(found)

if amount_found != 0:
        for (x, y, width, height) in found:
                cv2.rectangle(img_rgb, (x, y),
                                        (x + height, y + width),
                                        (0, 255, 0), 5)

plt.subplot(1, 1, 1)
plt.imshow(img_rgb)
plt.show()
```

Input:

output:



37. video in Reverse mode

Aim:

To play a video in Reverse mode using openv

Program:

```
#Reversing a Video
import cv2

cap = cv2.VideoCapture('video.mp4')
num_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
for i in reversed(range(num_frames)):
    cap.set(cv2.CAP_PROP_POS_FRAMES, i)
    ret, frame = cap.read()
    cv2.imshow('Reverse Video', frame)
    if cv2.waitKey(25) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

output:

38.Face Detection

Aim:

To detect a face from the given image by general Object recognition using OpenCV.

Program:

```
#Face detection
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
image = cv2.imread('1.jpg')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
cv2.imshow('Output', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input:

output:

39.To Detect vehicle in video

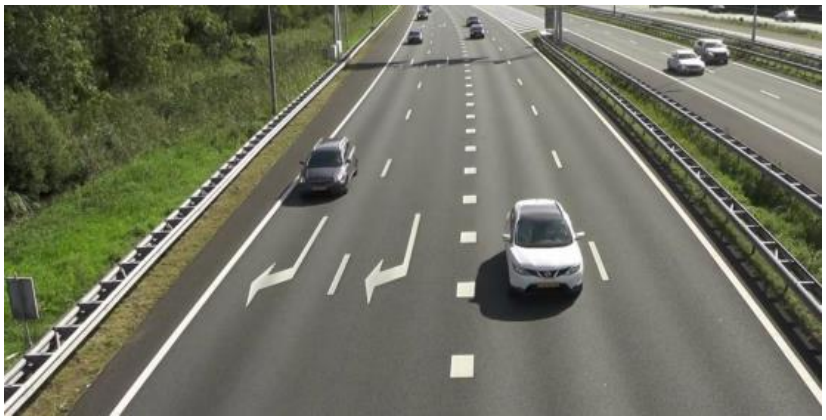
Aim:

To detect vehicle in a video,using openv

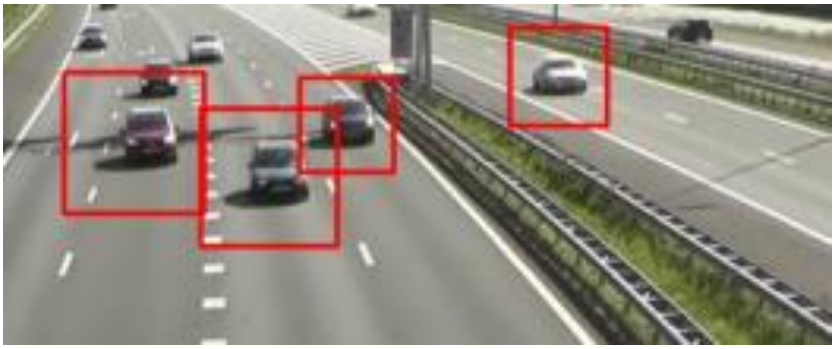
Program:

```
import cv2
cap = cv2.VideoCapture('F:/WhatsApp Video 2023-05-11 at 12.12.02.mp4')
car_cascade = cv2.CascadeClassifier('haarcascade_car.xml')
while True:
    ret, frame = cap.read()
    if not ret:
        break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cars = car_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
    for (x, y, w, h) in cars:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)
    cv2.imshow('Vehicle Detection', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Input:



Output:



40. Object recognition(watch)

Aim:

To Recognize object (watch) from the given image by general Object recognition using OpenCV.

Program:

```
#Draw Rectangular shape and extract objects
import cv2

img = cv2.imread('1.jpg')
start_point = (50, 50)
end_point = (200, 200)
color = (0, 0, 255)
thickness = 2
rect_img = cv2.rectangle(img, start_point, end_point, color, thickness)
cv2.imshow('Image with Rectangle', rect_img)
cv2.waitKey(0)
obj_img = img[start_point[1]:end_point[1], start_point[0]:end_point[0]]
cv2.imshow('Extracted Object', obj_img)
cv2.waitKey(0)
cv2.imwrite('object.jpg', obj_img)
cv2.destroyAllWindows()
```

Input:

output: