SIMATS SCHOOL OF ENGINEERING

NAME:- G.S.L.S.MANOGNA

REG-NO:-192124076

DSA0202-COMPUTER VISION WITH OPENCV FOR PATTREN RECOGNITION

LAB ACTIVITY :-

1)Perform basic Image Handling and processing operations on the image.

AIM:- Read an image in python and Convert an Image to Grayscale

PROGRAM:-

```
import cv2
img = cv2.imread ('C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg')
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
cv2.imwrite('gray_image.jpg', gray_img)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

2) Perform basic Image Handling and processing operations on the image

AIM:- Read an image in python and Convert an Image to Blur using GaussianBlur.

PROGRAM:-

```
import cv2
image = cv2.imread('C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg')
blur = cv2.GaussianBlur(image, (5, 5), 10)
cv2.imwrite('blurred_image.jpg', blur)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

3) Perform basic Image Handling and processing operations on the image

AIM:-

Read an image in python and Convert an Image to show outline using Canny function.

PROGRAM:-

```
import cv2
img = cv2.imread('C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 100, 200)
cv2.imwrite('Canny_Edges_3rd.jpg', edges)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

4) Perform basic Image Handling and processing operations on the image

AIM:-

Read an image in python and Dilate an Image using Dilate function.

PROGRAM:-

```
import cv2

import numpy as np

img = cv2.imread('C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg',
cv2.IMREAD_GRAYSCALE)

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

dilated_img = cv2.dilate(img, kernel, iterations=1)

cv2.imwrite('Dilated_Image.jpg', dilated_img)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

5. Perform basic Image Handling and processing operations on the image

AIM:-

Read an image in python and Erode an Image using erode function

PROGRAM:-

```
import cv2
import numpy as np
img = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg")
gray_image = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (5, 5))
eroded_image = cv2.erode(gray_image, kernel, iterations=1)
cv2.imwrite("eroded_image.jpg",eroded_image)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

6. Perform basic video processing operations on the captured video

AIM:- Read captured video in python and display the video, in slow motion and in fast motion.

PROGRAM:-

```
import cv2

cap = cv2.VideoCapture('C:\\Users\\ghant\\Desktop\\demonslayer.mp4')

slow_factor = 0.5

fast_factor = 2.0

ret, frame = cap.read()

while ret:

    cv2.imshow('Slow Motion', cv2.resize(frame, None, fx=slow_factor, fy=slow_factor))

    cv2.imshow('Fast Motion', cv2.resize(frame, None, fx=fast_factor, fy=fast_factor))

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

        break

    ret, frame = cap.read()

cap.release()

cv2.destroyAllWindows()
```

#TO RUN THIS CODE" video/demonslayer.mp4 at main · Muttamatam-Sreeharsha-0471/video (github.com)"DOWNLOAD VIDEO VIA THIS LINK

RESULT:-

Thus the program is executed successfully.

7)

AIM:-Capture video from web Camera and Display the video, in slow motion and in fast motion. operations on the captured video

PROGRAM:-

```
import cv2
cap = cv2.VideoCapture('C:\\Users\\ghant\\Desktop\\demonslayer.mp4')
slow_factor = 0.5
fast_factor = 2.0
ret, frame = cap.read()
while ret:
    cv2.imshow('Slow Motion', cv2.resize(frame, None, fx=slow_factor, fy=slow_factor))
    cv2.imshow('Fast Motion', cv2.resize(frame, None, fx=fast_factor, fy=fast_factor))
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
    ret, frame = cap.read()
cap.release()
cv2.destroyAllWindows()
```

RESULT:-

Thus the program is executed successfully.

8. Scaling an image to its Bigger and Smaller sizes.

AIM:-Converting an image to image to its Bigger and Smaller sizes.

PROGRAM:-

```
import cv2
image = cv2.imread("C:\\Users\\ghant\\OneDrive\\Desktop\\shin.jpg")
height, width = image.shape[:2]

scale_factor = 3.0
bigger_image = cv2.resize(image,(int(width*scale_factor),
int(height*scale_factor)))

scale_factor = 0.5
smaller_image = cv2.resize(image,(int(width*scale_factor),
int(height*scale_factor)))

cv2.imshow('original image',image)
cv2.imshow('Bigger image',bigger_image)
cv2.imshow('smaller image',smaller_image)

cv2.imwrite("Bigger_image.jpg",bigger_image)
cv2.imwrite("smaller_image.jpg",smaller_image)
```

INPUT:-

OUTPUT:-

Bigger image:-

Smaller image:-

RESULT:-

Thus the program is executed successfully.

9. Perform Rotation of an image to clockwise and counter clockwise direction.

AIM:-

Rotation of an image to clockwise and counter clockwise direction.

PROGRAM:-

```
import cv2
img = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg")
# Rotate clockwise
rotated_img = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
# Rotate counterclockwise
rotated_img = cv2.rotate(img, cv2.ROTATE_90_COUNTERCLOCKWISE)
cv2.imwrite("rotated_image.jpg",rotated_img)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

10. Perform moving of an image from one place to another.

AIM:- Moving of an image from one place to another.

PROGRAM:-

```
import cv2

image = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg")

width = image.shape[1]

height = image.shape[0]

new_image = cv2.imread("C:\\Users\\sreeh\\Desktop\\OpenCV\\image1.jpg")

current_position = cv2.getWindowProperty("Original Image",
cv2.WND_PROP_POSITION)

cv2.moveWindow("Original Image", current_position[0] + 100,
current_position[1] + 100)

cv2.imshow("Original Image", image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:-

The image will move from one position to another.

RESULT:-

Thus the program is executed successfully.

11. Perform Affine Transformation on the image.

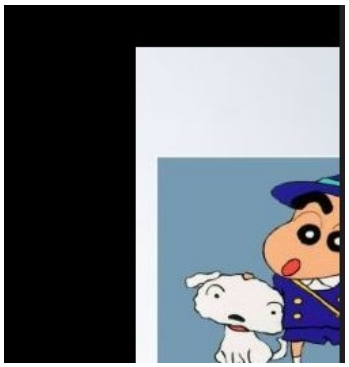
AIM:- Affine Transformation on the image

PROGRAM:-

```
import cv2
import numpy as np
img = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg")
rows, cols = img.shape[:2]
M = np.float32([[1, 0, 1000], [0, 1, 500]])
affine_img = cv2.warpAffine(img, M, (cols, rows))
cv2.imwrite('Affine_Transformed.jpg', affine_img)
```

INPUT:-

OUTPUT:-



RESULT:-

Thus the program is executed successfully.

12. Perform Perspective Transformation on the image.

AIM:- Perspective Transformation on the image.

PROGRAM:-

```
import cv2
import numpy as np
img = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg")
rows, cols = img.shape[:2]
src_points = np.float32([[0, 0], [cols - 1, 0], [0, rows - 1], [cols - 1, rows - 1]])
dst_points = np.float32([[0, 0], [cols - 1, 0], [int(0.33*cols), rows - 1],
[int(0.66*cols), rows - 1]])
M = cv2.getPerspectiveTransform(src_points, dst_points)
perspective_img = cv2.warpPerspective(img, M, (cols, rows))
cv2.imwrite('Perspective_Transformed_Image.jpg', perspective_img)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

13. Perform Perspective Transformation on the Video.

AIM:- Perspective Transformation on the Video

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.astype(np.float32))
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()
```

RESULT:-

It will access the cam and transform your own live video

14. Perform transformation using Homography matrix.

AIM:- transformation using Homography matrix

PROGRAM:-

```
import cv2

import numpy as np

img = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg")

rows, cols = img.shape[:2]

src_points = np.float32([[0, 0], [cols - 1, 0], [0, rows - 1], [cols - 1, rows - 1]])

dst_points = np.float32([[0, 0], [cols - 1, 0], [0, int(0.7*rows)], [cols - 1,
int(0.7*rows)]])

M, _ = cv2.findHomography(src_points, dst_points)

homography_img = cv2.warpPerspective(img, M, (cols, rows))

cv2.imwrite('transformation_using_Homography_Image.jpg',
homography_img)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

15. Perform transformation using Direct Linear Transformation.

AIM:- Transformation using Direct Linear Transformation.

PROGRAM:-

```
import cv2
import numpy as np

image = cv2.imread("C:\\Users\\ghant\\OneDrive\\Desktop\\shin.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)
```

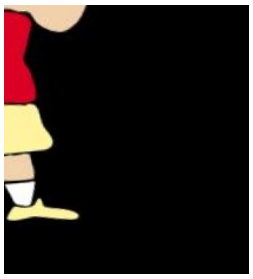
```
h = V[-1, :]
homography_matrix = h.reshape((3, 3))

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. Perform Edge detection using canny method

AIM:- Edge detection using canny method

PROGRAM:-

```
import cv2
import numpy as np
img = cv2.imread('C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg',0)
edges = cv2.Canny(img,100,200)
```

cv2.imwrite('Edges.jpg',edges)

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

17. Perform Edge detection using Sobel Matrix along X axis

AIM:- Edge detection using Sobel Matrix along X axis

PROGRAM:-

```
import cv2
import numpy as np
img = cv2.imread('C:\\Users\\ghant\\OneDrive\\Desktop\\image1.jpg',0)
sobel_x = cv2.Sobel(img,cv2.CV_8U,1,0,ksize=5)
cv2.imwrite('sobel_x.jpg',sobel_x)
```

INPUT:

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

18. Perform Edge detection using Sobel Matrix along Y axis

AIM:- Edge detection using Sobel Matrix along Y axis

PROGRAM:-

```
import cv2
import numpy as np
img = cv2.imread('C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg',0)
sobel_y = cv2.Sobel(img,cv2.CV_8U,0,1,ksize=5)
cv2.imwrite('sobel_y.jpg',sobel_y)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

19. Perform Edge detection using Sobel Matrix along XY axis

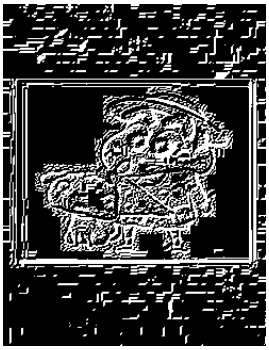
AIM:- Edge detection using Sobel Matrix along XY axis

PROGRAM:-

```
import cv2
import numpy as np
img = cv2.imread(' C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg', 0)
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=3)
sobely = cv2.Sobel(img, cv2.CV_64F, 0, 1, ksize=3)
edges = cv2.addWeighted(sobelx, 0.5, sobely, 0.5, 0)
cv2.imwrite('Edge_detection.jpg', edges)
```

INPUT:-

OUTPUT:-



RESULT:-

Thus the program is executed successfully.

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

0	1	0
1	−4	1
0	1	0

AIM:- Sharpening of Image using Laplacian mask with negative center coefficient.

PROGRAM:-

```
import cv2

import numpy as np
```

```
img = cv2.imread('C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg')
kernel = np.array([[0,1,0], [1,-4,1], [0,1,0]])
sharpened = cv2.filter2D(img, -1, kernel)
cv2.imwrite('Sharpened_Image.jpg', sharpened)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

21. Perform Sharpening of Image using Laplacian mask implemented with an extension of diagonal neighbors.

1	1	1
1	−8	1
1	1	1

AIM:- Perform Sharpening of Image using Laplacian mask implemented with an extension of diagonal neighbors.

PROGRAM:-

```
import cv2
import numpy as np
img = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg'")
kernel = np.array([[1,1,1], [1,-8,1], [1,1,1]])
sharpened = cv2.filter2D(img, -1, kernel)
cv2.imwrite('Sharpened_Image.jpg', sharpened)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

22. Perform Sharpening of Image using Laplacian mask with positive center coefficient.

Mask of Laplacian + addition

$$
\begin{aligned}
g(x,y) &= f(x,y) - [f(x+1,y) + f(x-1,y) \\
&\quad + f(x,y+1) + f(x,y-1) + 4f(x,y)] \\
&= 5f(x,y) - [f(x+1,y) + f(x-1,y) \\
&\quad + f(x,y+1) + f(x,y-1)]
\end{aligned}
$$

0	-1	0
-1	5	-1
0	-1	0

AIM:- Sharpening of Image using Laplacian mask with positive center coefficient.

PROGRAM:-

```
import cv2
import numpy as np
image = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.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.jpg', sharpened)
```

INPUT:-

OUTPUT:-

RESULT:-Thus the program is executed successfully.

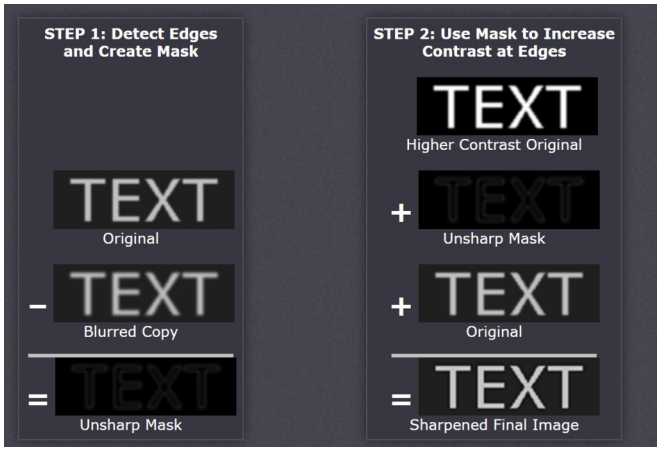
23. Perform Sharpening of Image using unsharp masking.

Unsharp masking

$$f_s(x,y) = f(x,y) - \bar{f}(x,y)$$

sharpened image = original image – blurred image

- to subtract a blurred version of an image produces sharpening output image.



AIM:- Sharpening of Image using unsharp masking.

PROGRAM:-

```
from PIL import Image, ImageFilter
image_path = "C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg"
im1 = Image.open(image_path)
im2 = im1.filter(ImageFilter.UnsharpMask(radius=3, percent=200, threshold=5))
```

im2.show()

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

24. Perform Sharpening of Image using High-Boost Masks.

High-boost Masks

0	−1	0	−1	−1	−1
−1	$A+4$	−1	−1	$A+8$	−1
0	−1	0	−1	−1	−1

- $A \geq 1$
- if A = 1, it becomes “standard” Laplacian sharpening

AIM:- Sharpening of Image using High-Boost Masks.

PROGRAM:-

```
import cv2
import numpy as np
def high_boost_filter(image, boost_factor):
    kernel_size = 3
```

```
    blur_factor = (kernel_size - 1) / 2
    kernel = np.ones((kernel_size, kernel_size), dtype=np.float32) /
(kernel_size ** 2)
    blur_image = cv2.filter2D(image, -1, kernel)
    mask = image + (image - blur_image) * boost_factor
    return mask
image_path = "C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg"
image = cv2.imread(image_path)
sharpened_image = high_boost_filter(image, 1.5)
cv2.imwrite('sharpened_image.jpg', sharpened_image)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

25. Perform Sharpening of Image using Gradient masking.

−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

AIM:- Sharpening of Image using Gradient masking

PROGRAM:-

```
import cv2

import numpy as np

image_path = "C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg"

image = cv2.imread(image_path)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

gradient_x = cv2.Sobel(gray, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=3)

gradient_y = cv2.Sobel(gray, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=3)

gradient = cv2.subtract(gradient_x, gradient_y)

cv2.imwrite('sharpened_image3.jpg', gradient)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

26. Insert water marking to the image using OpenCV.

AIM:- Insert water marking to the image using OpenCV

PROGRAM:-

```
import cv2
import math
logo = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg'")
img = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image2.jpg'")
h_logo, w_logo, _ = logo.shape
h_img, w_img, _ = img.shape
center_y = int(h_img/2)
center_x = int(w_img/2)
top_y = center_y - int(h_logo/2)
left_x = center_x - int(w_logo/2)
bottom_y = top_y + h_logo
right_x = left_x + w_logo
destination = img[top_y:bottom_y, left_x:right_x]
result = cv2.addWeighted(destination, 1, logo, 0.5, 0)
img[top_y:bottom_y, left_x:right_x] = result
```

```
cv2.imshow("watermarked.jpg", img)
cv2.imshow("Watermarked Image", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

INPUT:-

Img1:-

Img2:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

27. Do Cropping, Copying and pasting image inside another image using OpenCV.

AIM:- Cropping, Copying and pasting image inside another image using OpenCV.

PROGRAM:-

#For cropping

```
import cv2
img = cv2.imread(' C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg ')
# Crop the image from (x, y) = (10, 10) to (x, y) = (300, 300)
crop_img = img[10:300, 10:300]
cv2.imshow("cropped", crop_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

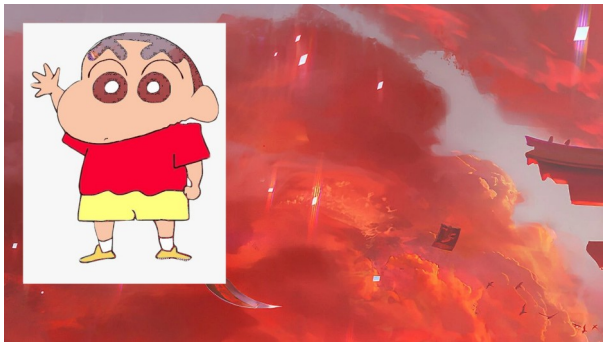
#For copy and paste

```
import cv2
img1 = cv2.imread(“C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg”)
img2 = cv2.imread(“C:\\Users\\ghant\\Desktop\\OpenCV\\image2.jpg”)
rows, cols, channels = img2.shape
roi = img1[50:50+rows, 50:50+cols]
img2gray = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)
ret, mask = cv2.threshold(img2gray, 10, 255, cv2.THRESH_BINARY)
mask_inv = cv2.bitwise_not(mask)
img1_bg = cv2.bitwise_and(roi, roi, mask=mask_inv)
img2_fg = cv2.bitwise_and(img2, img2, mask=mask)
dst = cv2.add(img1_bg, img2_fg)
img1[50:50+rows, 50:50+cols] = dst
cv2.imshow('result', img1)
cv2.waitKey(0)
```

INPUT:-

OUTPUT:- (for cropped)

(for copy and paste)



RESULT:-

Thus the program is executed successfully.

28. Find the boundary of the image using Convolution kernel for the given image.

AIM:- The boundary of the image using Convolution kernel for the given image.

PROGRAM:-

import cv2

```
import numpy as np
img = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.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('convolutional Image', sharpened_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

29. Morphological operations based on OpenCV using Erosion technique.

AIM:- Morphological operations based on OpenCV using Erosion technique.

PROGRAM:-

```
import cv2
import numpy as np
img = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.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:-

RESULT:-

Thus the program is executed successfully.

30. Morphological operations based on OpenCV using Dilation technique.

AIM:- Morphological operations based on OpenCV using Dilation technique.

PROGRAM:-

```
import cv2

import numpy as np

img = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.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:-

RESULT:-

Thus the program is executed successfully.

31. Morphological operations based on OpenCV using Opening technique.

AIM:-Do morphological operations in an image

PROGRAM:-

```
import cv2
import numpy as np
img = cv2.imread('C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg')
kernel = np.ones((5,5),np.uint8)
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
cv2.imwrite('Opened.jpg', opening)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

32. Morphological operations based on OpenCV using Closing technique.

AIM:- Morphological operations using Closing technique.

PROGRAM:-

```
import cv2

import numpy as np

img = cv2.imread('C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg',
cv2.IMREAD_GRAYSCALE)

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

closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel)

cv2.imwrite('Closing.jpg', closing)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

33. Morphological operations based on OpenCV using Morphological Gradient technique.

AIM:- Morphological operations using Morphological Gradient technique.

PROGRAM:-

```
import cv2
import numpy as np
img = cv2.imread('C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg', 0)
kernel = np.ones((3,3), np.uint8)
gradient = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)
cv2.imwrite('Morphological_Gradient.jpg', gradient)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

34. Morphological operations based on OpenCV using Top hat technique.

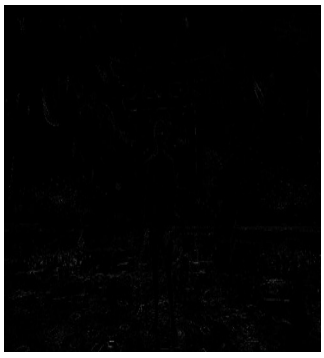
AIM:- Morphological operations using Top hat technique.

PROGRAM:-

```
import cv2
filterSize =(3, 3)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,filterSize)
input_image = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg")
input_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)
tophat_img = cv2.morphologyEx(input_image,cv2.MORPH_TOPHAT,kernel)
cv2.imwrite("tophat.jpg", tophat_img)
```

INPUT:-

OUTPUT:-



RESULT:-

Thus the program is executed successfully.

35. Morphological operations based on OpenCV using Black hat technique.

AIM:- Morphological operation using Black hat technique.

PROGRAM:-

```
import cv2
filterSize =(3, 3)
kernel = cv2.getStructuringElement(cv2.MORPH_RECT,filterSize)
input_image = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.jpg")
input_image = cv2.cvtColor(input_image, cv2.COLOR_BGR2GRAY)
blackhat_img =
cv2.morphologyEx(input_image,cv2.MORPH_BLACKHAT,kernel)
cv2.imwrite("blackhat.jpg", blackhat_img)
```

INPUT:-

OUTPUT:-

RESULT:-

Thus the program is executed successfully.

36. Recognise watch from the given image by general Object recognition using OpenCV.

AIM:- Recognise watch from the given image by general Object recognition

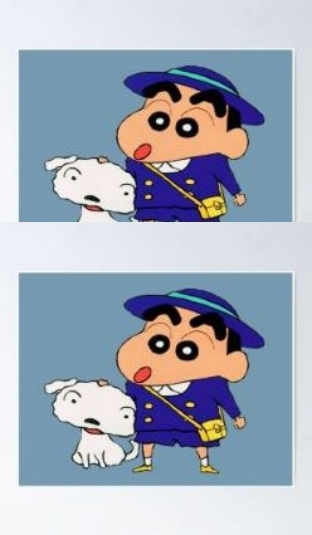
PROGRAM:-

```
import cv2
# Load the image
img = cv2.imread('watch.jpg')

# Load the model
model =
cv2.dnn.readNetFromTensorflow('frozen_inference_graph.pb',
'ssd_mobilenet_v2_coco_2018_03_29.pbtxt')
# Set the input image and scale factor
input_blob = cv2.dnn.blobFromImage(img, size=(300, 300),
swapRB=True, crop=False)
model.setInput(input_blob)
# Run the detection and get the output
output = model.forward()
# Loop through the detected objects and draw rectangles around the
watches
for detection in output[0,0,:,:]:
    confidence = detection[2]
    if confidence > 0.5:
        left = int(detection[3] * img.shape[1])
        top = int(detection[4] * img.shape[0])
        right = int(detection[5] * img.shape[1])
        bottom = int(detection[6] * img.shape[0])
        cv2.rectangle(img, (left, top), (right, bottom), (0, 255, 0), 3)
```

```
# Display the image with detections
cv2.imshow('Image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

INPUT:-



OUTPUT:-

RESULT:-

Thus the program is executed successfully.

37. Using Opencv play Video in Reverse mode.

AIM:- Play Video in Reverse mode

PROGRAM:-

```
import cv2
cap = cv2.VideoCapture("C:\\Users\\ghant\\Desktop\\demonslayer.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()
```

INPUT:-

#TO RUN THIS CODE" video/demonslayer.mp4 at main · Muttamatam-Sreeharsha-0471/video (github.com)"DOWNLOAD VIDEO VIA THIS LINK

OUTPUT:-

The video will run in reverse mode.

RESULT:-

Thus the program is executed successfully.

38. Face Detection using Opencv.

AIM:- Face Detection

PROGRAM:-

```
import cv2
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)
while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
    cv2.imshow('Face Recognition', frame)
```

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

OUTPUT:-

It will access the cam and detect the face.

RESULT:-

Thus the program is executed successfully.

39. Vehicle Detection in a Video frame using OpenCV .

AIM:- Vehicle Detection in a Video frame

PROGRAM:-

```
import cv2
# Load the video file
cap = cv2.VideoCapture('video.mp4')
# Load the vehicle detection model
car_cascade = cv2.CascadeClassifier('haarcascade_car.xml')
# Loop through the video frames
while True:
    # Read a frame from the video
    ret, frame = cap.read()
    # Stop the loop if there are no more frames
    if not ret:
        break
    # Convert the frame to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    # Detect vehicles in the frame
    cars = car_cascade.detectMultiScale(gray, scaleFactor=1.1,
minNeighbors=5)
    # Draw rectangles around the detected vehicles
    for (x, y, w, h) in cars:
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 0, 255), 2)
```

```
    # Display the frame with the vehicle detections
    cv2.imshow('Vehicle Detection', frame)
    # Exit the loop if the 'q' key is pressed
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
# Release the video capture and destroy all windows
cap.release()
cv2.destroyAllWindows()
```

RESULT:-

Thus the program is executed successfully.

40. **Draw Rectangular shape and extract objects**.

AIM:- Draw Rectangular shape and extract objects

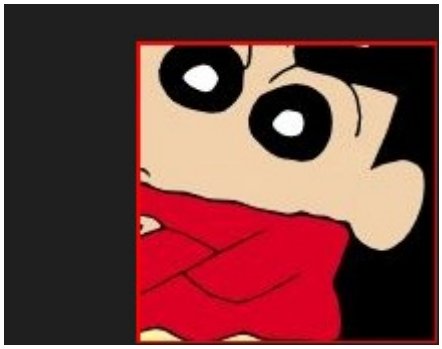
PROGRAM:-

```
import cv2
img = cv2.imread("C:\\Users\\ghant\\Desktop\\OpenCV\\image1.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:-



RESULT:-

Thus the program is executed successfully.