SIMATS SCHOOL OF ENGINEERING

DSA0201 – COMPUTER VISION WITH OPENCV FOR IMAGE PROCESSING

NAME: DIVYA.S

REGISTER NUMBER: 192124069

1. Perform basic Image Handling and processing operations on the image. • Read an image in python and Convert an Image to Grayscale

AIM:

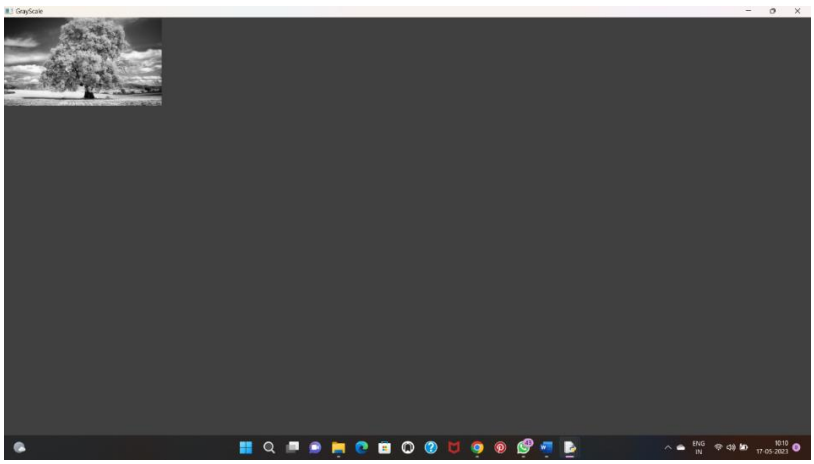
To Perform Basic Operations to Read Image and Convert to Grayscale using Python

PROGRAM:

```
import cv2

import numpy as np

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

print(kernel)

path = "C:/Users/Welcome/OneDrive/Pictures/Saved Pictures/cat.jpeg"

img =cv2.imread(path)

imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

cv2.imshow("GrayScale",imgGray)

cv2.waitKey(0)
```

INPUT:

OUTPUT:



2. Perform basic Image Handling and processing operations on the image.• Read an image in python and Convert an Image to Blur using GaussianBlur.

AIM:

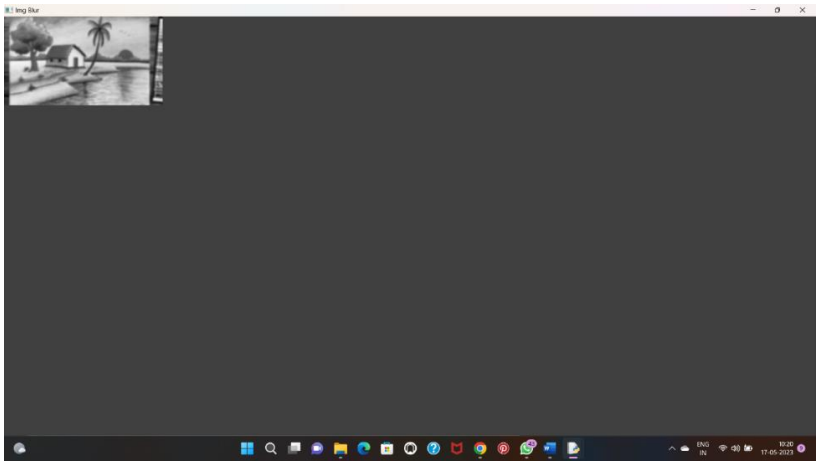
To Perform Basic Operations to Read Image and Convert to Blur using GaussianBlur.

PROGRAM

```
import cv2
import numpy as np
kernel = np.ones((5,5),np.uint8)
print(kernel)
path = "C:/Users/divya/OneDrive/Documents/COMPUTER VISION/Picture2.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)
```


INPUT:

OUTPUT:



3. Perform basic Image Handling and processing operations on the image• Read an image in python and Convert an Image to show outline using Canny function

AIM:

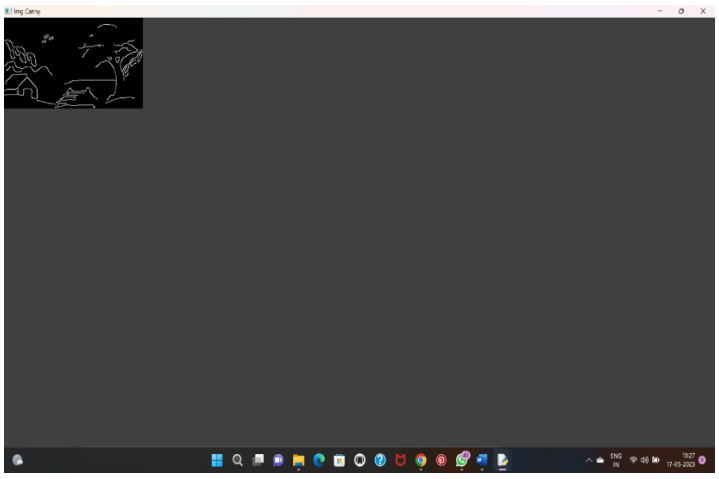
To Perform Basic Operations to Convert image to show outline Canny function in Python

PROGRAM:

```
import cv2

import numpy as np

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

print(kernel)

path = "C:/Users/divya/OneDrive/Documents/COMPUTER VISION/CANNY.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)

cv2.imshow("Img Canny",imgCanny)

cv2.waitKey(0)
```


INPUT:

OUTPUT:



4. Perform basic Image Handling and processing operations on the image• Read an image in python and Dilate an Image using Dilate function

AIM:

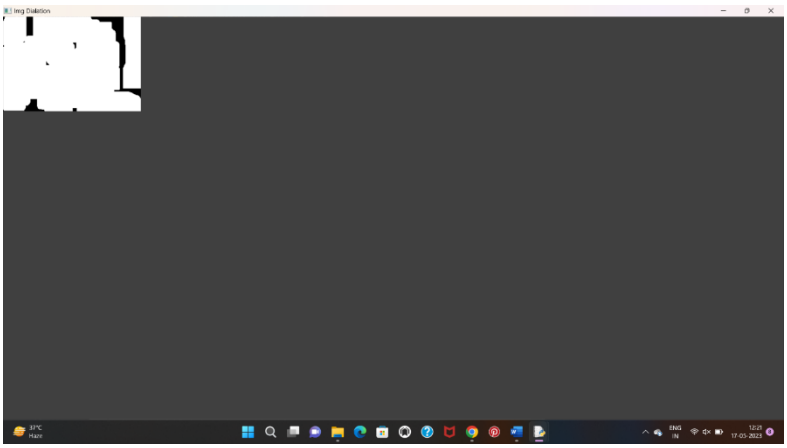
To Perform Basic Operations to Read Image and Dilate an Image using Python

PROGRAM:

```
import cv2

import numpy as np

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

print(kernel)

path = "C:/Users/divya/OneDrive/Documents/COMPUTER VISION/dialation.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("Img Erosion",imgEroded)

cv2.waitKey(0)
```


INPUT:

OUTPUT:



5. Perform basic Image Handling and processing operations on the image• Read an image in python and Erode an Image using erode function

AIM:

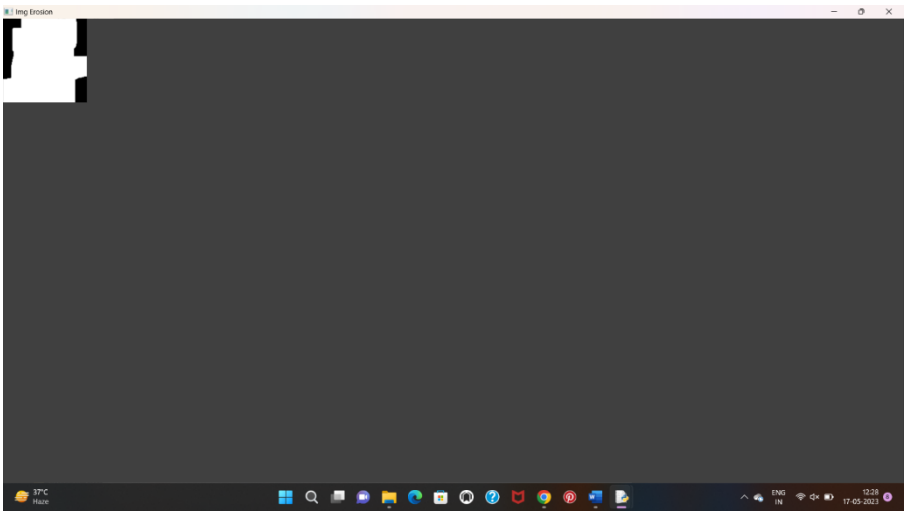
The Aim of the experiment is to Read an image in python and Erode an Image using erode function

PROGRAM:

```
import cv2

import numpy as np

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

print(kernel)

path = "C:/Users/divya/OneDrive/Documents/COMPUTER VISION/erosion.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("Img Erosion",imgEroded)

cv2.waitKey(0)
```


INPUT:

OUTPUT:



6. Perform basic video processing operations on the captured video• Read captured video in python and display the video, in slow motion and in fast motion.

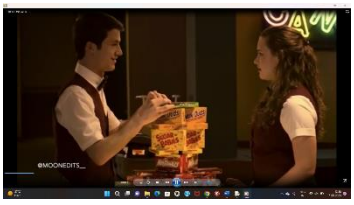
AIM:

The Aim of the Experiment is to Read captured video in python and display the video, in slow motion and in fast motion

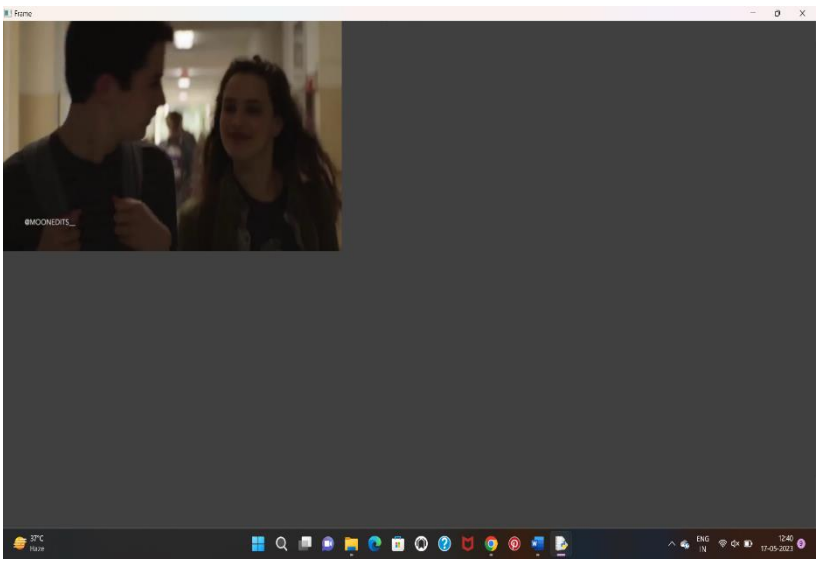
PROGRAM:

```
import cv2

import numpy as np

cap = cv2.VideoCapture("C:/Users/divya/OneDrive/Documents/COMPUTER VISION/13 REASONS WHY")

if (cap.isOpened()== False):

        print("Error opening video file")

while(cap.isOpened()):

        ret, frame = cap.read()

        if ret == True:

                cv2.imshow('Frame', frame)

                if cv2.waitKey(250) & 0xFF == ord('q'):

                        break

        else:

                break

cap.release()

cv2.destroyAllWindows()
```


INPUT:



OUTPUT:



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

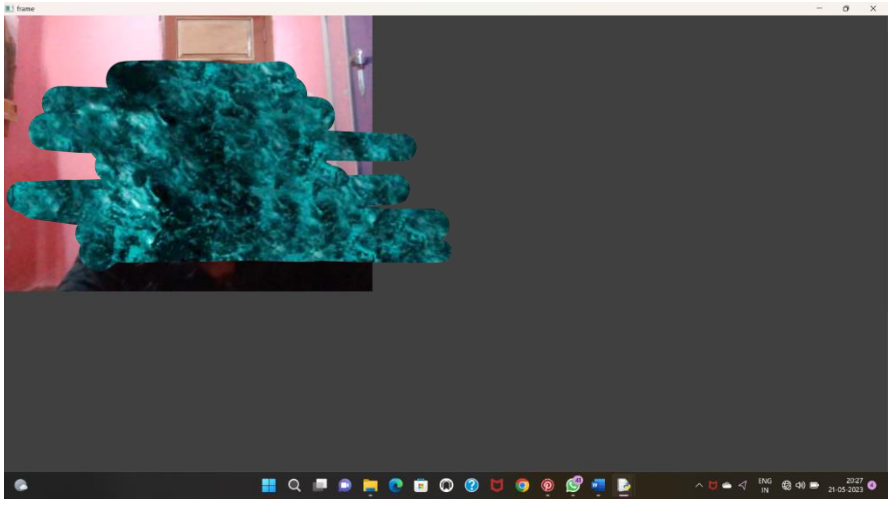
AIM:

The Aim is to 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(0)

height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))

fps = cap.get(cv2.CAP_PROP_FPS)

path = "0"

fourcc = cv2.VideoWriter_fourcc(*'mp4v')

output = cv2.VideoWriter(path, fourcc, 2,(width, height))

while True:

        ret, frame = cap.read()

        cv2.imshow("frame", frame)

        output.write(frame)

        k = cv2.waitKey(24)

        if k == ord("q"):

                break

cap.release()

output.release()

cv2.destroyAllWindows()
```


OUTPUT:



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

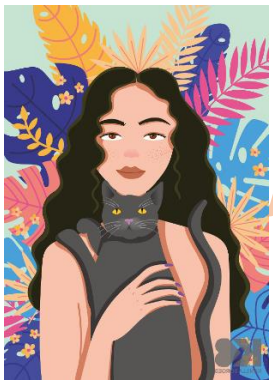
AIM:

The Aim is resize the image from bigger to smaller size

PROGRAM:

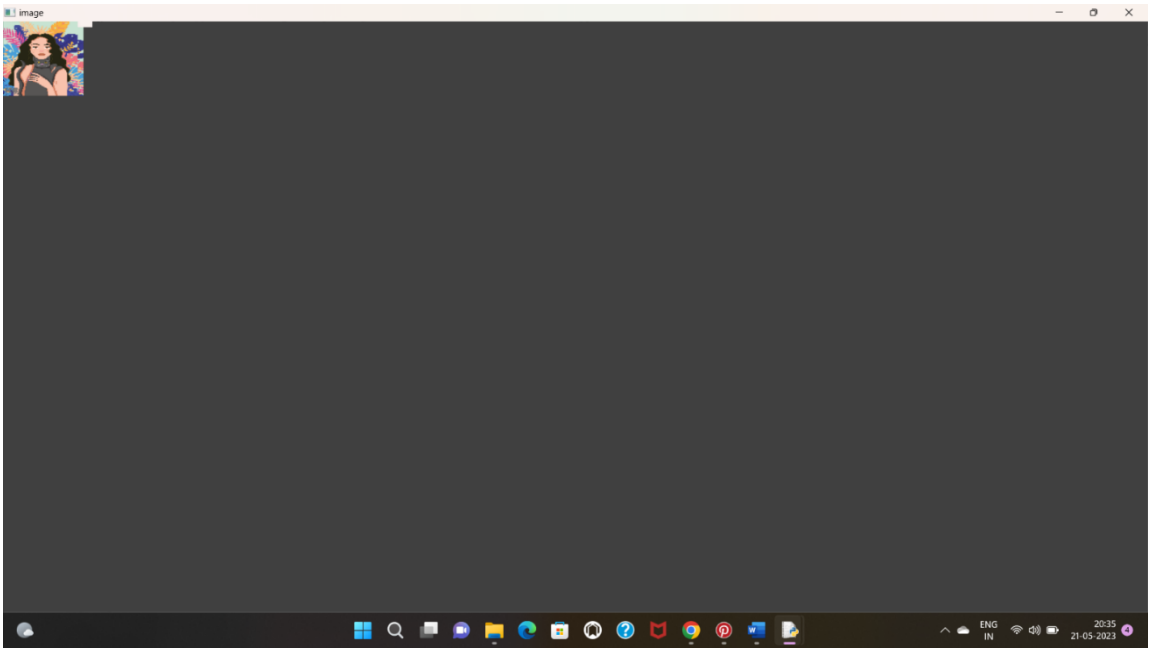
```
import cv2

import numpy as np

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

img = cv2.imread("C:/Users/Welcome/OneDrive/Pictures/Saved
Pictures/cat.jpeg",cv2.IMREAD_COLOR)

img = cv2.resize(img,(600,600))

cv2.imshow("image",img)

cv2.waitKey(0)
```

INPUT:



OUTPUT:

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



ROTATION 90 ALONG DEGREE:

AIM

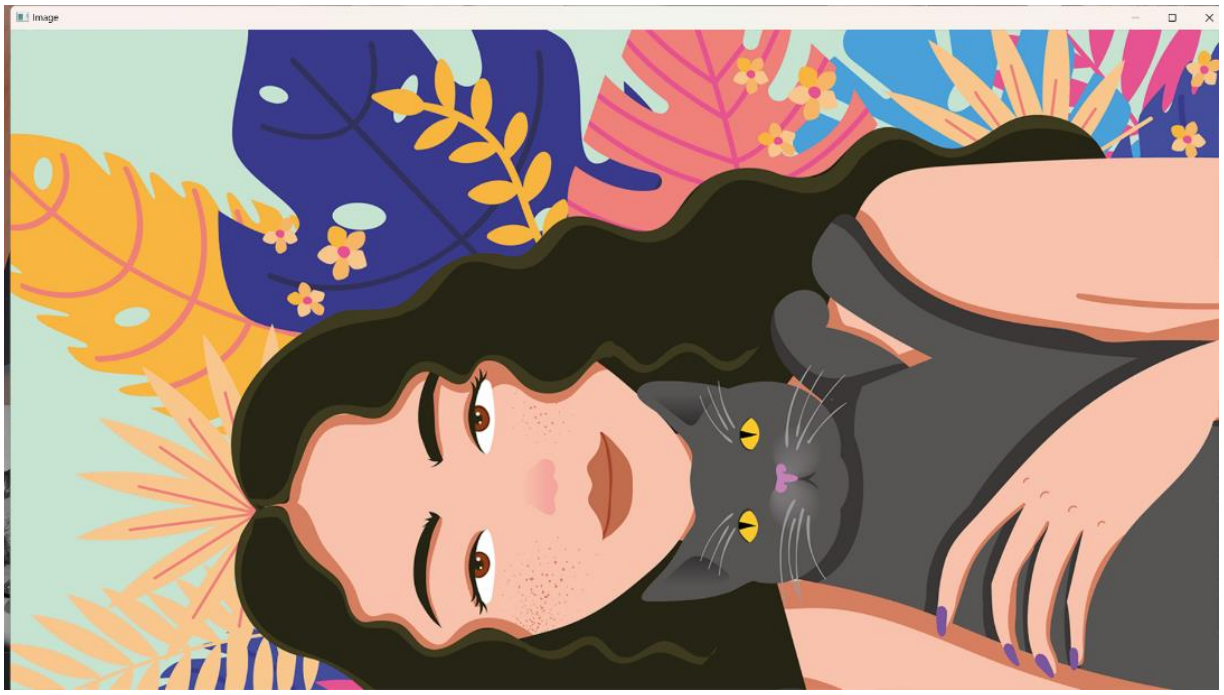
The Aim of the Experiment is to perform Rotation of an image along 90 degree

PROGRAM

```
import cv2
path r"C:/Users/Welcome/OneDrive/Pictures/SavedPictures/cat.jpeg"
 src = cv2.imread(path)
 window_name = 'Image'
image = cv2.rotate(src, cv2.ROTATE_180)
 cv2.imshow(window_name, image)
cv2.waitKey(0)
```

INPUT

OUTPUT



ROTATION ALONG 180 DEGREE

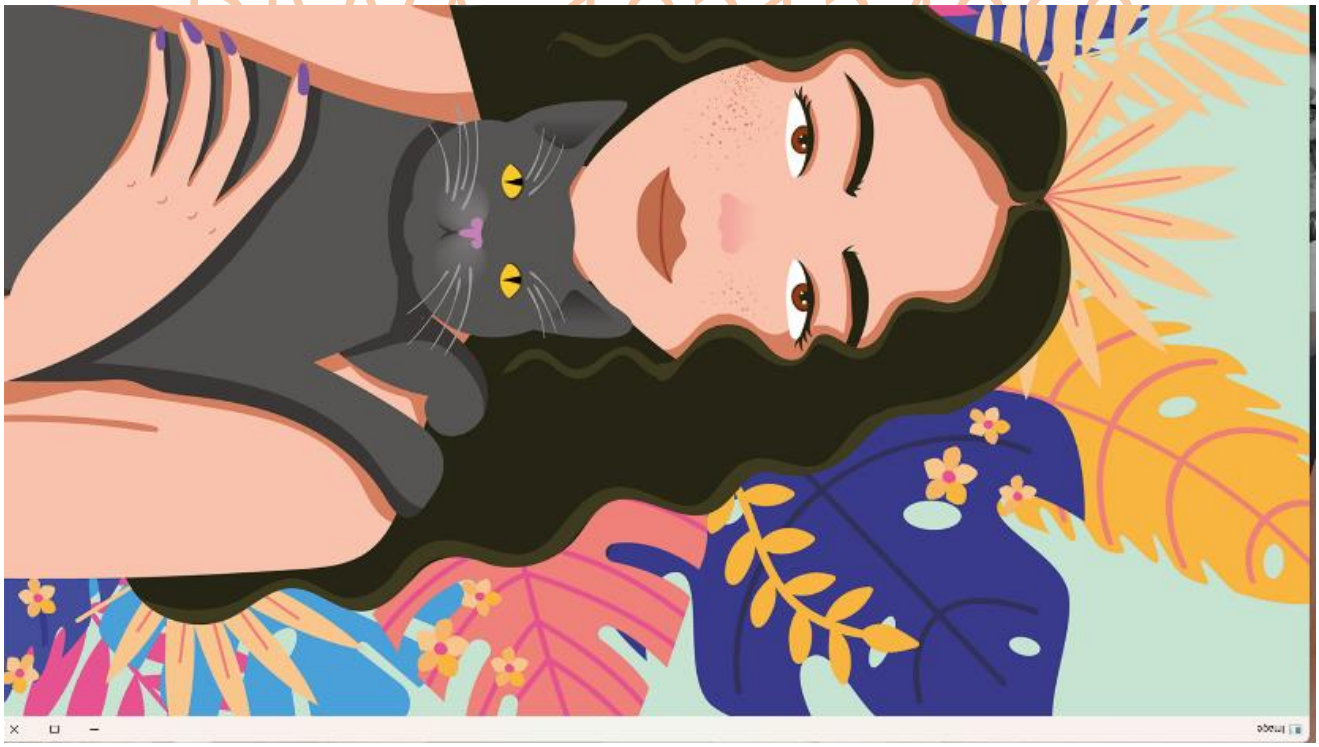
AIM

The Aim of the Experiment is to perform Rotation of an image along 180 degree

PROGRAM

```
import cv2
path = r"C:/Users/divya/OneDrive/Documents/COMPUTER VISION/Girl with a Cat.png"
src = cv2.imread(path)
 window_name = 'Image'
image = cv2.rotate(src, cv2.ROTATE_90_COUNTERCLOCKWISE)
 # Displaying the image
cv2.imshow(window_name, image)
cv2.waitKey(0)
```

OUTPUT



ROTATION ALONG 270 DEGREE

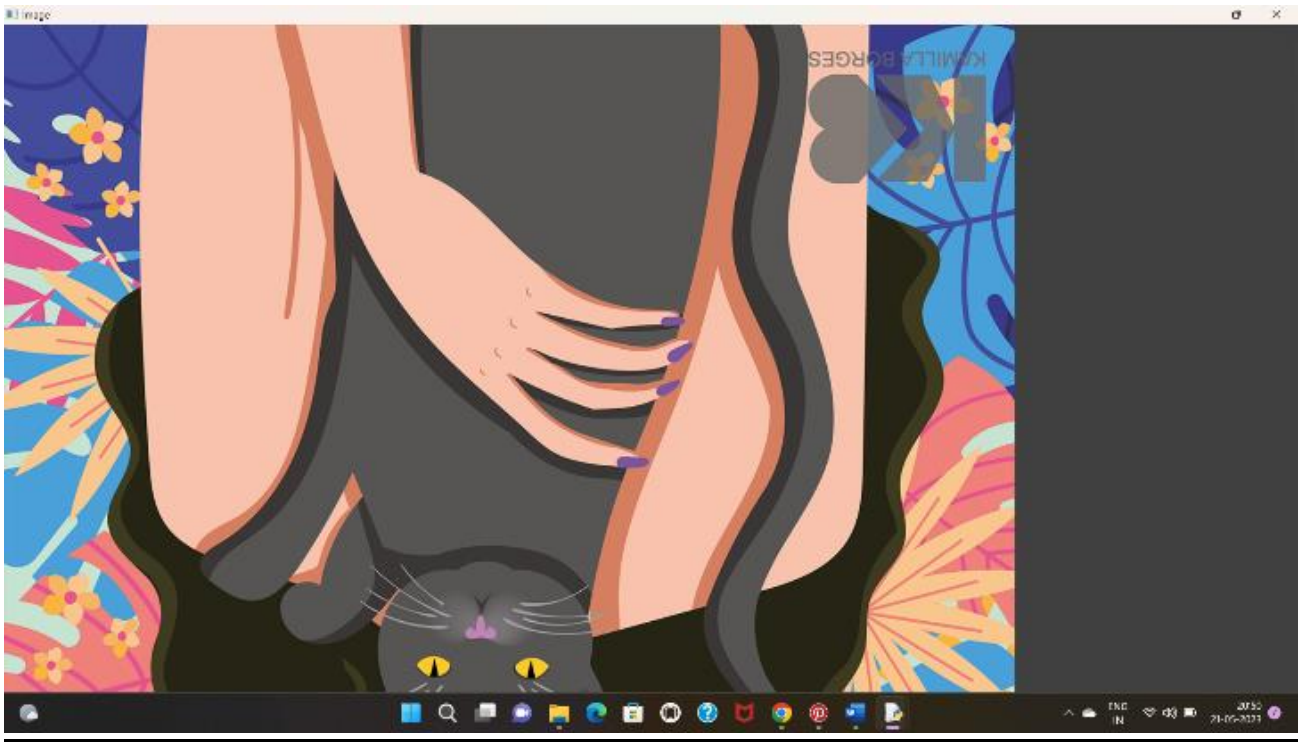
AIM

The Aim of the Experiment is to perform Rotation of an image along 270 degree

PROGRAM

```
import cv2
path = r"C:/Users/Welcome/OneDrive/Pictures/Saved Pictures/cat.jpeg"
 src = cv2.imread(path)
 window_name = 'Image'
image = cv2.rotate(src, cv2.ROTATE_90_COUNTERCLOCKWISE)
cv2.imshow(window_name, image)
cv2.waitKey(0)
```

OUTPUT

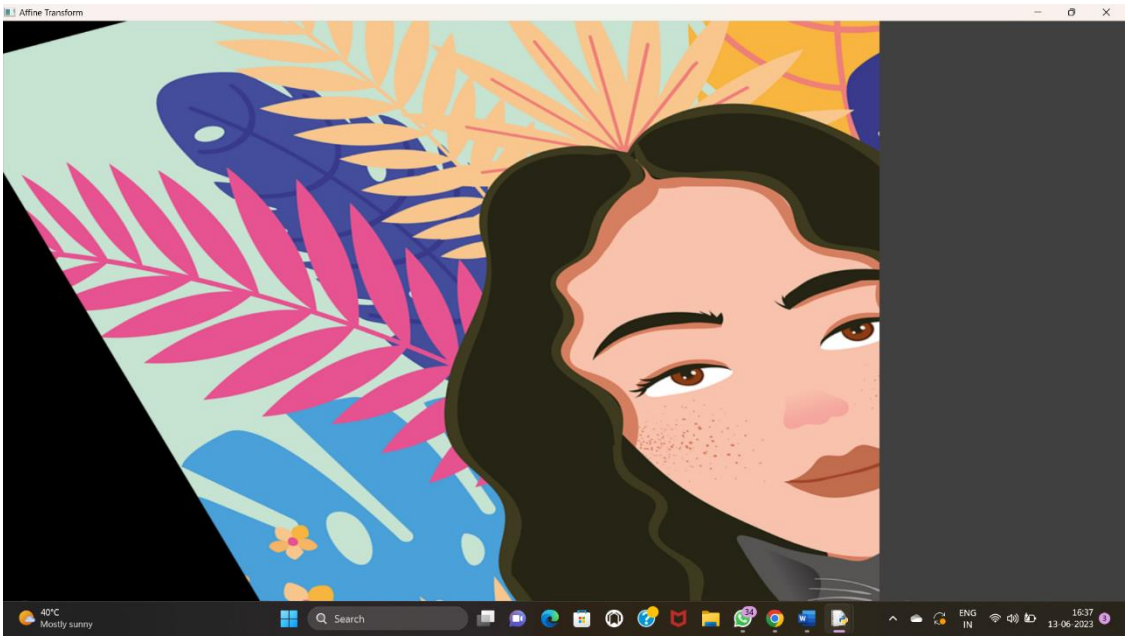


11. Perform Affine Transformation on the image.

PROGRAM:

```
import cv2
import numpy as np
# read the input image
img = cv2.imread("C:/Users/Welcome/OneDrive/Pictures/Saved Pictures/afiine.jpg")
# access the image height and width
rows,cols,_ = img.shape
# define at three point on input image
pts1 = np.float32([[50,50],[200,50],[50,200]])
# define three points corresponding location to output image
pts2 = np.float32([[10,100],[200,50],[100,250]])
# get the affine transformation Matrix
M = cv2.getAffineTransform(pts1,pts2)
# apply affine transformation on the input image
dst = cv2.warpAffine(img,M,(cols,rows))
cv2.imshow("Affine Transform", dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


OUTPUT

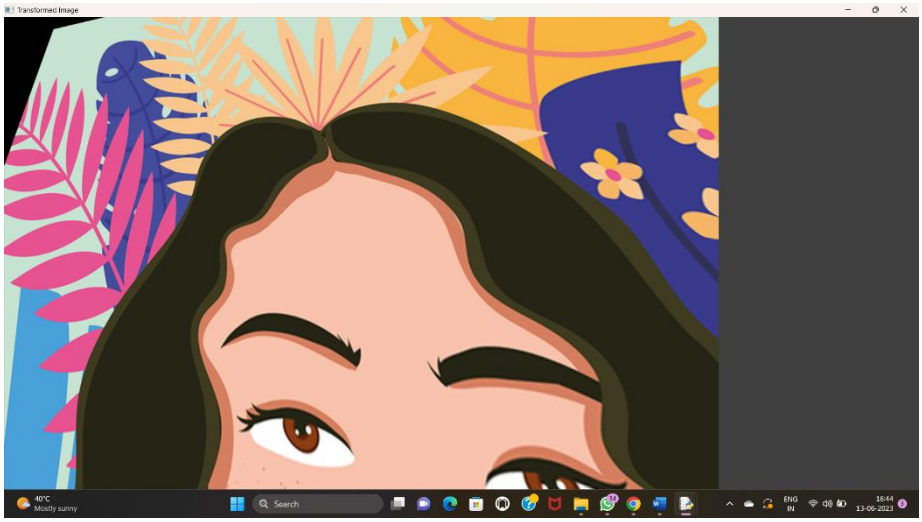


12. Perform Perspective Transformation on the image.

PROGRAM

```
# import required libraries
import cv2
import numpy as np
# read the input image
img = cv2.imread("C:/Users/Welcome/OneDrive/Pictures/Saved Pictures/afiine.jpg")
# find the height and width of image
# width = number of columns, height = number of rows in image array
rows,cols,ch = img.shape
# define four points on input image
pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]])
# define the corresponding four points on output image
pts2 = np.float32([[100,50],[300,0],[0,300],[300,300]])
# get the perspective transform matrix
M = cv2.getPerspectiveTransform(pts1,pts2)
# transform the image using perspective transform matrix
dst = cv2.warpPerspective(img,M,(cols, rows))
# display the transformed image
cv2.imshow('Transformed Image', dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


OUTPUT

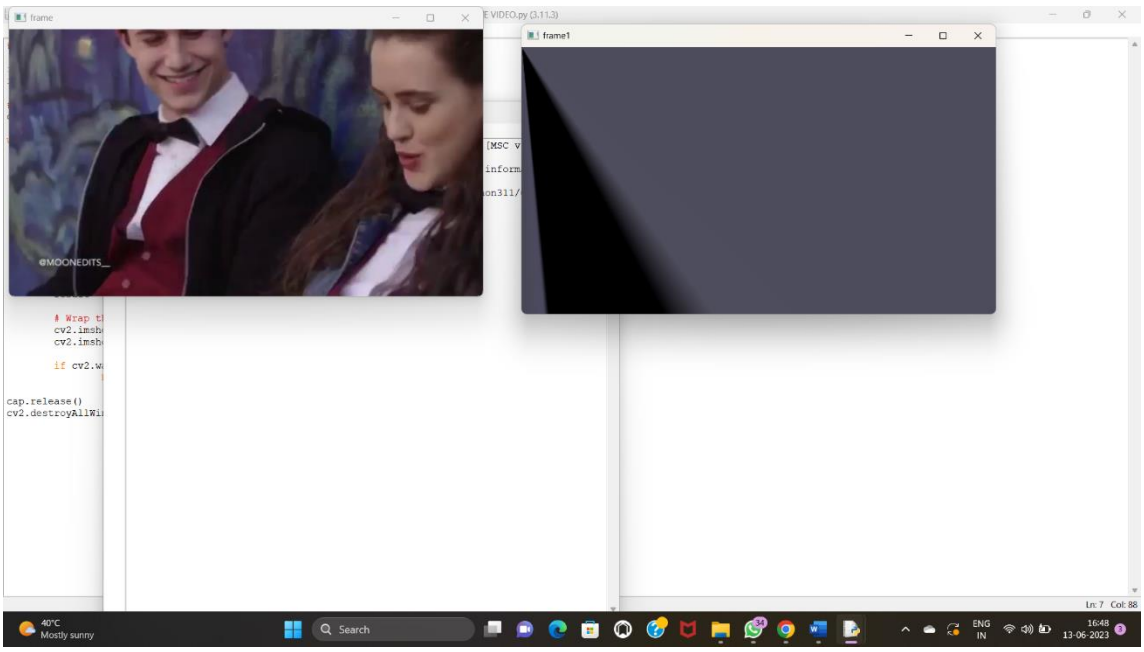


13. Perform Perspective Transformation on the Video.

PROGRAM:

```
import cv2

import numpy as np

cap = cv2.VideoCapture("C:/Users/Welcome/Downloads/pexels-pixabay-855029-1920x1080-60fps.mp4")

while True:

        ret, frame = cap.read()

       pts1 = np.float32([[200,300], [5, 2],

                                        [0, 4], [6, 0]])

        pts2 = np.float32([[0, 0], [4, 0],

                                        [0, 1], [4, 6]])

        matrix = cv2.getPerspectiveTransform(pts1, pts2)

        result = cv2.warpPerspective(frame, matrix, (0, 0))

        cv2.imshow('frame', frame) # Initial Capture

        cv2.imshow('frame1', result) # Transformed Capture

if cv2.waitKey(24) == 27:

                break

cap.release()

cv2.destroyAllWindows()
```


OUPUT:

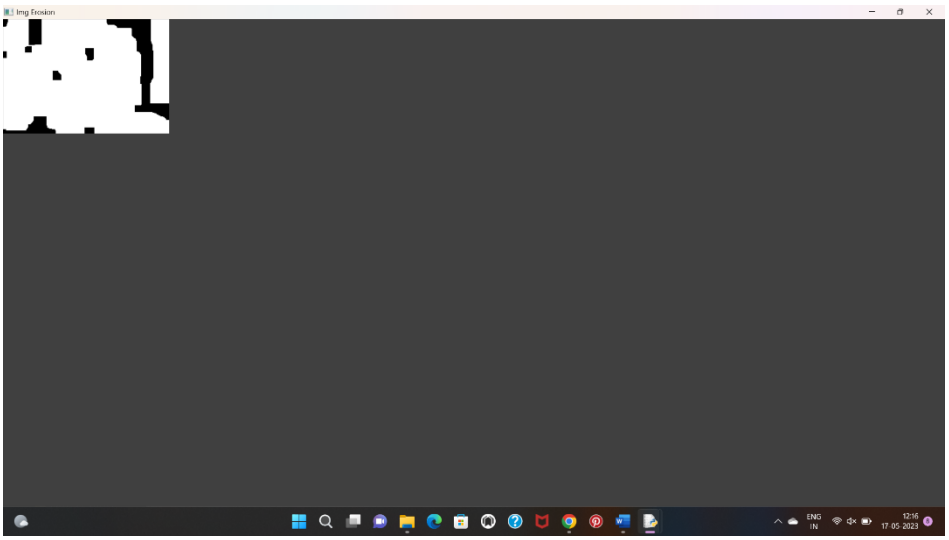


14. Perform transformation using Homography matrix

PROGRAM:

```
import cv2
import numpy as np
    # Read source image.
    im_src = cv2.imread("C:/Users/Welcome/OneDrive/Pictures/Saved Pictures/afiine.jpg")
    # Four corners of the book in source image
    pts_src = np.array([[141, 131], [480, 159], [493, 630],[64, 601]])
   # Read destination image.
    im_dst = cv2.imread("C:/Users/Welcome/OneDrive/Pictures/Saved Pictures/afiine.jpg")
    # Four corners of the book in destination image.
    pts_dst = np.array([[318, 256],[534, 372],[316, 670],[73, 473]])
    # Calculate Homography
    h, status = cv2.findHomography(pts_src, pts_dst)
   # Warp source image to destination based on homography
    im_out = cv2.warpPerspective(im_src, h, (im_dst.shape[1],im_dst.shape[0]))
    # Display images
    cv2.imshow("Source Image", im_src)
    cv2.imshow("Destination Image", im_dst)
    cv2.imshow("Warped Source Image", im_out)
    cv2.waitKey(0)
```


OUTPUT:

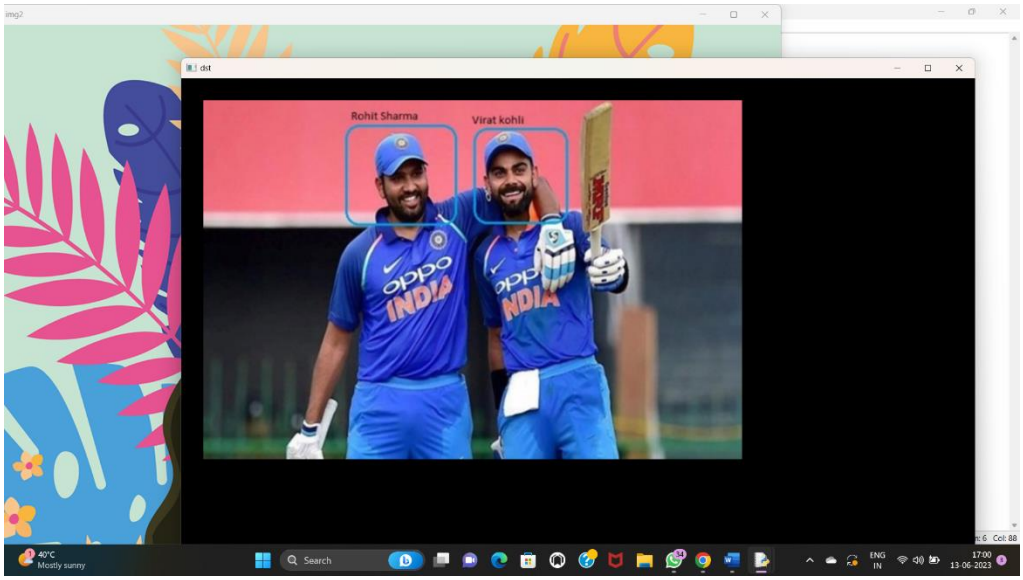


15. Perform transformation using Direct Linear Transformation

PROGRAM

```
import cv2
import numpy as np
# Load images
img1 = cv2.imread("C:/Users/Welcome/OneDrive/Pictures/Saved Pictures/afiine.jpg")
img2 = cv2.imread("C:/Users/Welcome/OneDrive/Pictures/Saved Pictures/cat.jpeg")
# Define corresponding points
pts1 = np.array([[50, 50], [200, 50], [50, 200], [200, 200]])
pts2 = np.array([[100, 100], [300, 100], [100, 300], [300, 300]])
# Estimate projective transformation matrix using DLT
H, _ = cv2.findHomography(pts1, pts2)
# Apply projective transformation to img1
dst = cv2.warpPerspective(img1, H, (img2.shape[1], img2.shape[0]))
# Display images
cv2.imshow('img1', img1)
cv2.imshow('img2', img2)
cv2.imshow('dst', dst)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


OUTPUT

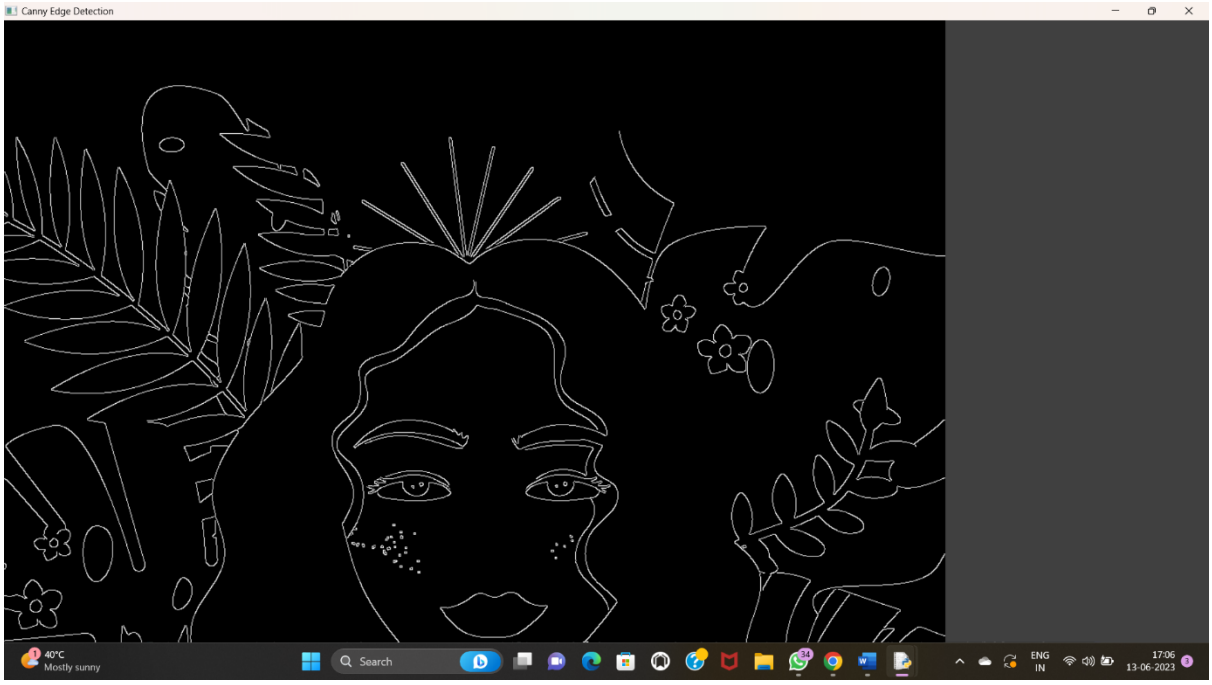


16. Perform Edge detection using canny method

PROGRAM:

```
import cv2
 img = cv2.imread("C:/Users/Welcome/OneDrive/Pictures/Saved Pictures/cat.jpeg")
cv2.imshow('Original', img)
cv2.waitKey(0)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
 edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200) # Canny Edge Detection
cv2.imshow('Canny Edge Detection', edges)
cv2.waitKey(0)
 cv2.destroyAllWindows()
```

OUPUT:

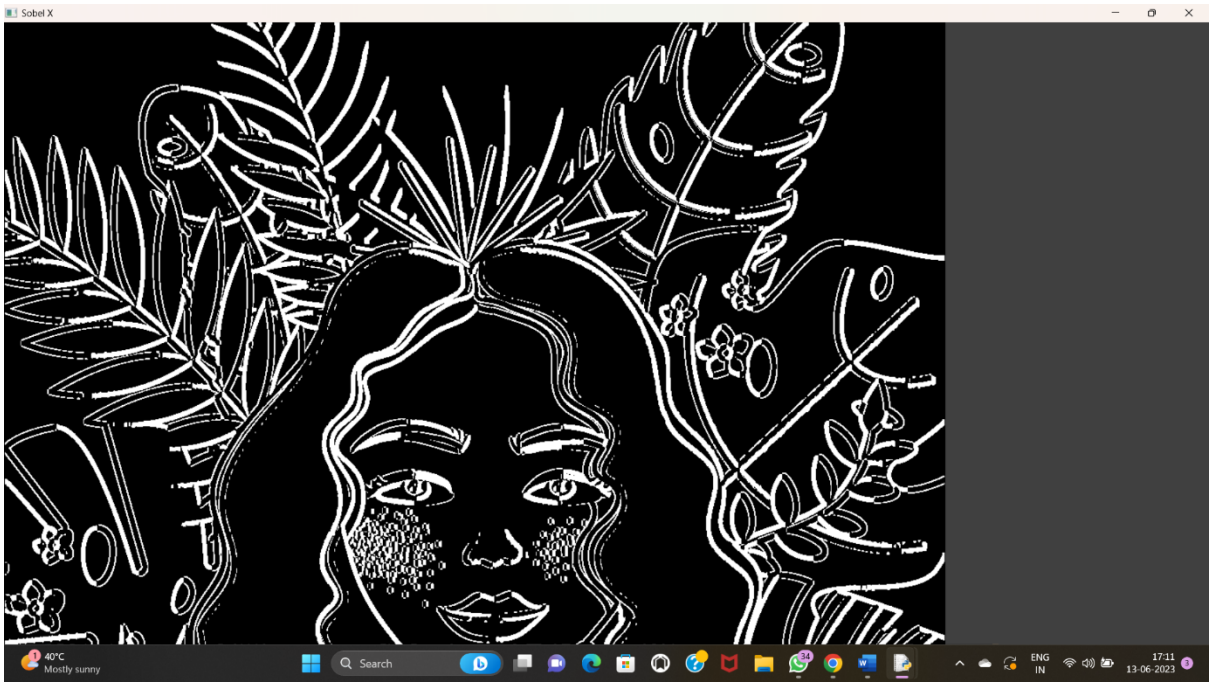


17. Perform Edge detection using Sobel Matrix along X axis

PROGRAM:

```
import cv2
 img = cv2.imread("C:/Users/Welcome/OneDrive/Pictures/Saved Pictures/cat.jpeg")
cv2.imshow('Original', img)
cv2.waitKey(0)
 img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Blur the image for better edge detection
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
 sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5) # Sobel Edge Detection on the X axis
cv2.imshow('Sobel X', sobelx)
cv2.waitKey(0)
```

OUTPUT:

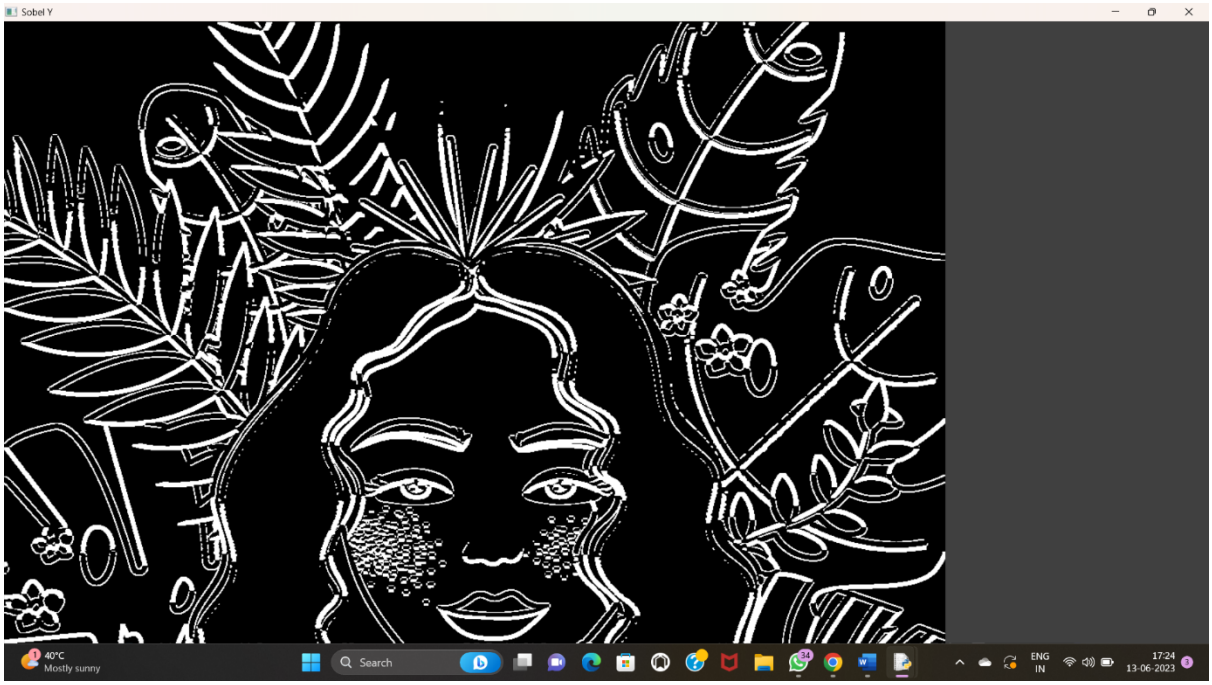


18. Perform Edge detection using Sobel Matrix along Y axis

PROGRAM:

```
import cv2
 # Read the original image
img = cv2.imread("C:/Users/Welcome/OneDrive/Pictures/Saved Pictures/cat.jpeg")
# Display original image
cv2.imshow('Original', img)
cv2.waitKey(0)
 # Convert to graycsale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Blur the image for better edge detection
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
 # Sobel Edge Detection
sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5) # Sobel Edge Detection on the Y axis
# Display Sobel Edge Detection Images
cv2.imshow('Sobel Y', sobely)
cv2.waitKey(0)
```


OUTPUT:

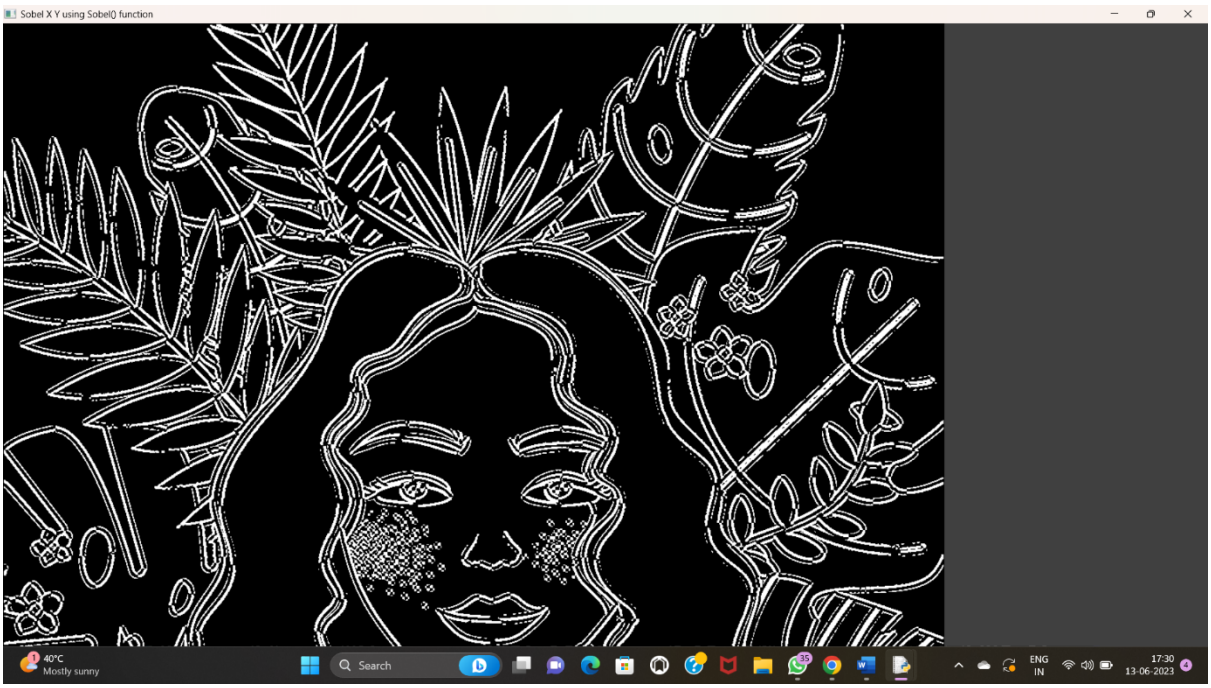


19. Perform Edge detection using Sobel Matrix along XY axis

PROGRAM

```
import cv2
 img = cv2.imread("C:/Users/Welcome/OneDrive/Pictures/Saved Pictures/cat.jpeg")
# Display original image
cv2.imshow('Original', img)
cv2.waitKey(0)
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Blur the image for better edge detection
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)
 sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5) # Combined X and Y Sobel Edge Detection
cv2.imshow('Sobel X Y using Sobel() function', sobelxy)
cv2.waitKey(0)
```


OUTPUT:

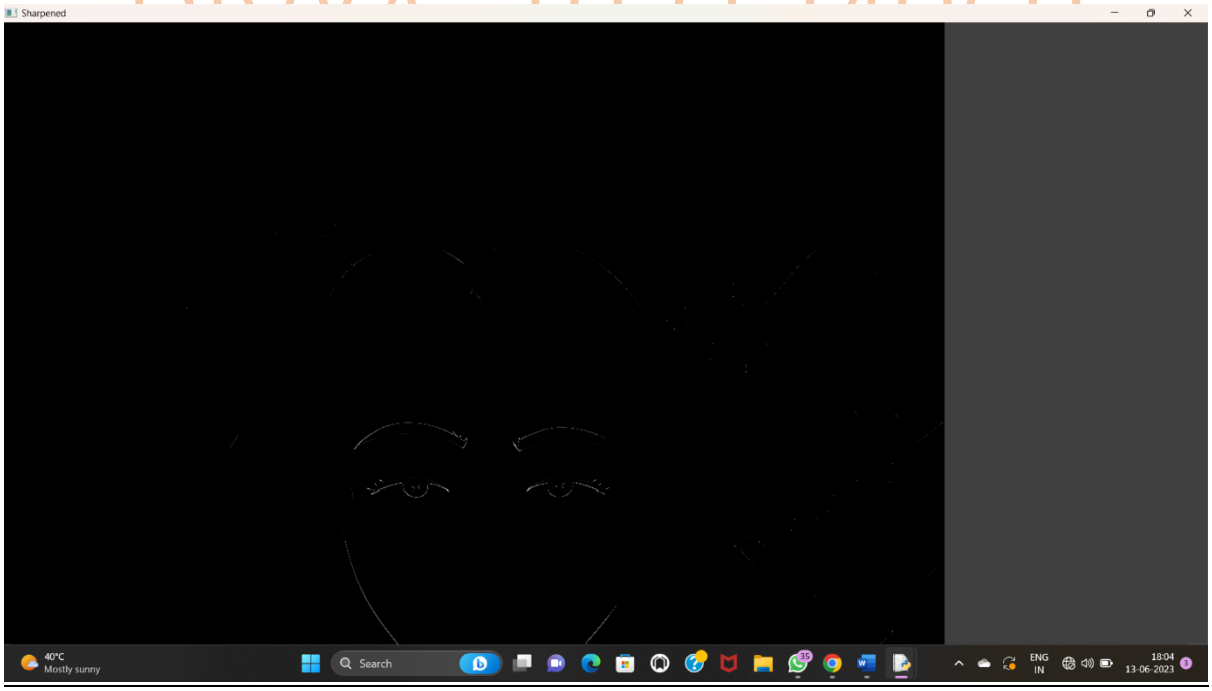


20. Perform Sharpening of Image using Laplacian mask with negative center coefficient.

PROGRAM:

```
import cv2
import numpy as np
img = cv2.imread("C:/Users/DIVYA/OneDrive/Pictures/ss.png")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
kernel = np.array([[0,1,0], [1,-8,1], [0,1,0]])
sharpened = cv2.filter2D(gray,-1, kernel)
cv2.imshow('Original', gray)
cv2.imshow('Sharpened', sharpened)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:

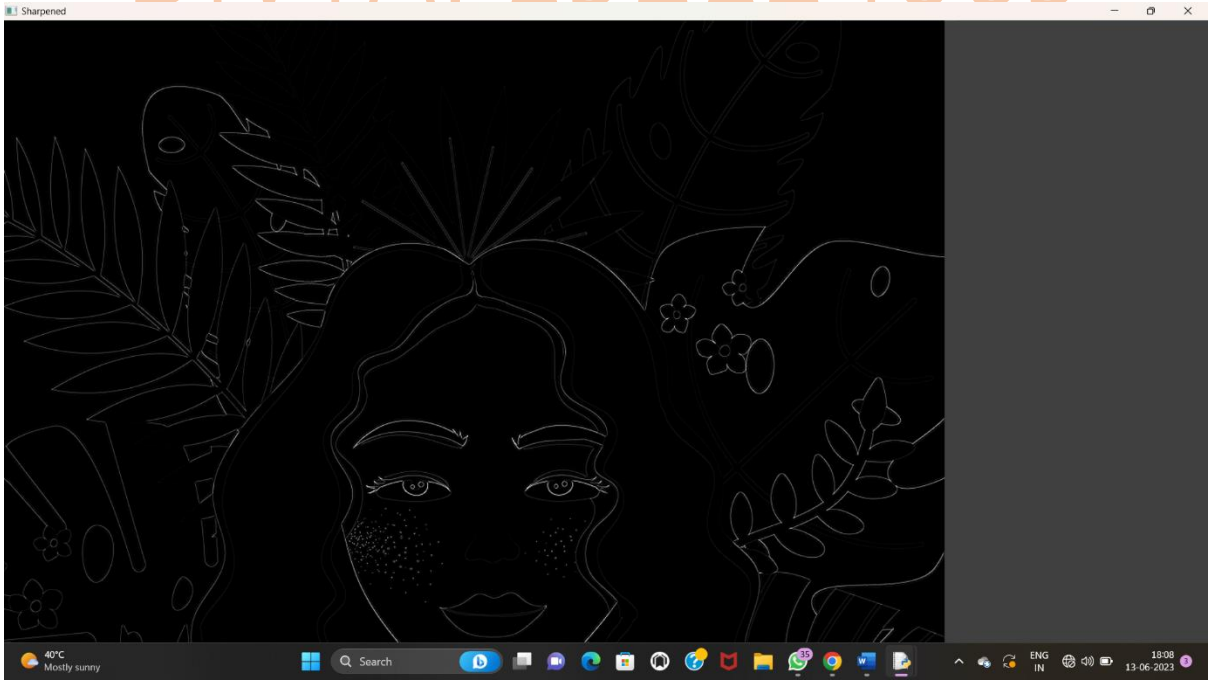


21. Perform Sharpening of Image using Laplacian mask implemented with an extension of diagonal neighbors,

PROGRAM

```
import cv2
import numpy as np
img = cv2.imread("C:/Users/Dama Prasoona/OneDrive/Pictures/21.png")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
kernel = np.array([[0,1,0], [1,-4,1], [0,1,0]])
sharpened = cv2.filter2D(gray,-1, kernel)
cv2.imshow('Original', gray)
cv2.imshow('Sharpened', sharpened)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


OUTPUT

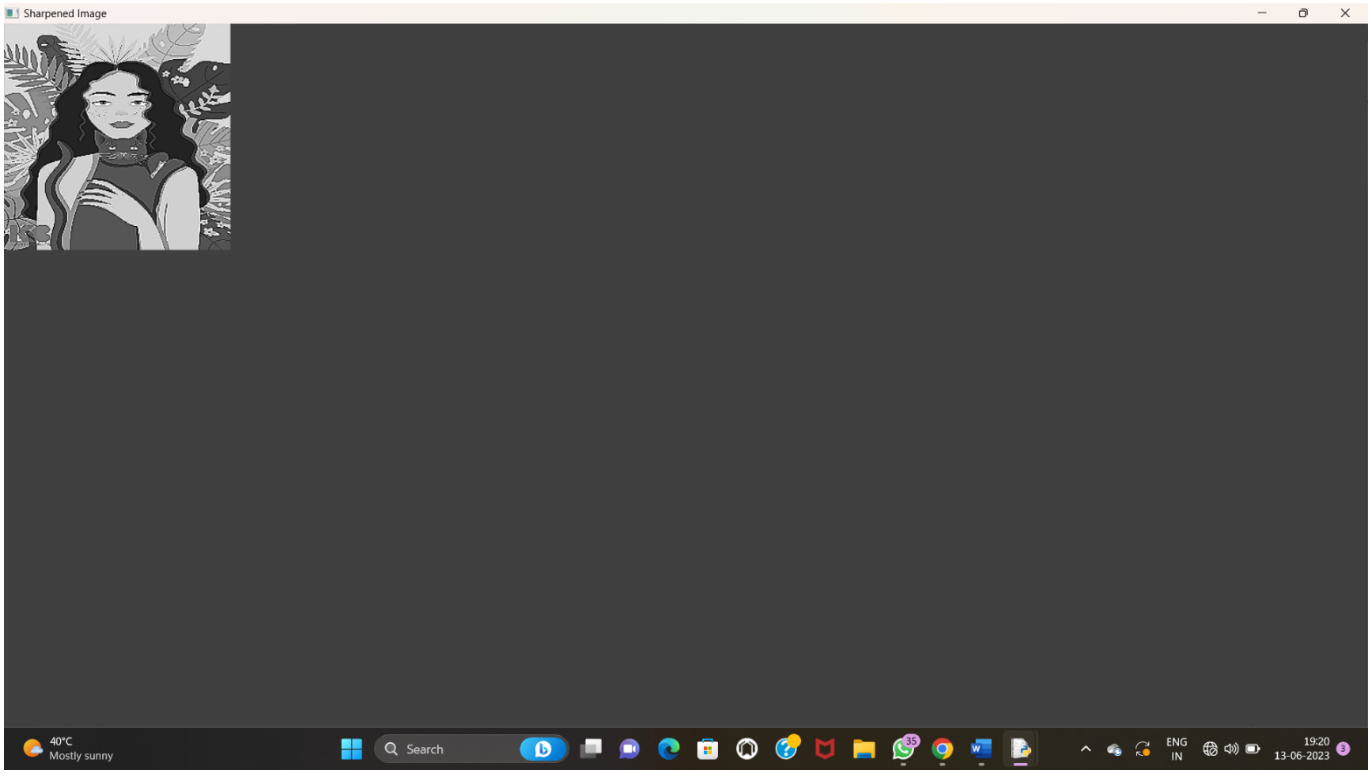


22. Perform Sharpening of Image using Laplacian mask with positive center coefficient.

PROGRAM:

```
import cv2
import numpy as np
img = cv2.imread("C:/Users/divya/Downloads/Girl with a Cat.png")
img = cv2.resize(img,(255, 255))
gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Apply the Laplacian filter with a positive center coefficient
laplacian_kernel = np.array([[0,-1, 0], [-1, 5,-1], [0,-1, 0]])
sharpened_img = cv2.filter2D(gray_img,-1, laplacian_kernel)
sharpened_img = cv2.cvtColor(sharpened_img, cv2.COLOR_GRAY2BGR)
cv2.imshow('Original Image', img)
cv2.imshow('Sharpened Image', sharpened_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


OUTPUT:

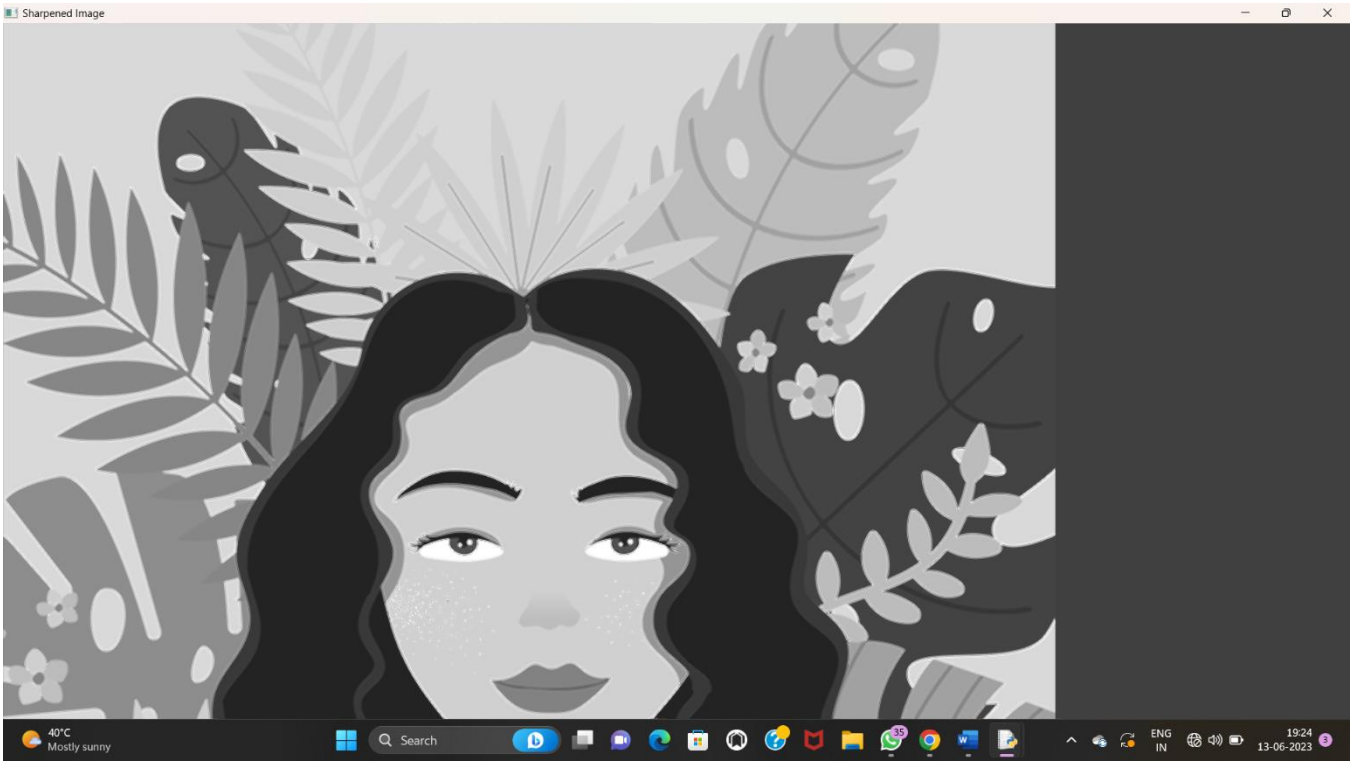


23. Perform Sharpening of Image using unsharp masking.

PROGRAM:

```
import cv2
import numpy as np
img = cv2.imread("C:/Users/divya/Downloads/Girl with a Cat.png")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
laplacian_kernel = np.array([[0, 1, 0],
                             [1,-4, 1],
                             [0, 1, 0]])
laplacian = cv2.filter2D(gray,-1, laplacian_kernel)
sharpened = cv2.add(gray, laplacian)
cv2.imshow('Original Image', gray)
cv2.imshow('Sharpened Image', sharpened)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


OUTPUT:

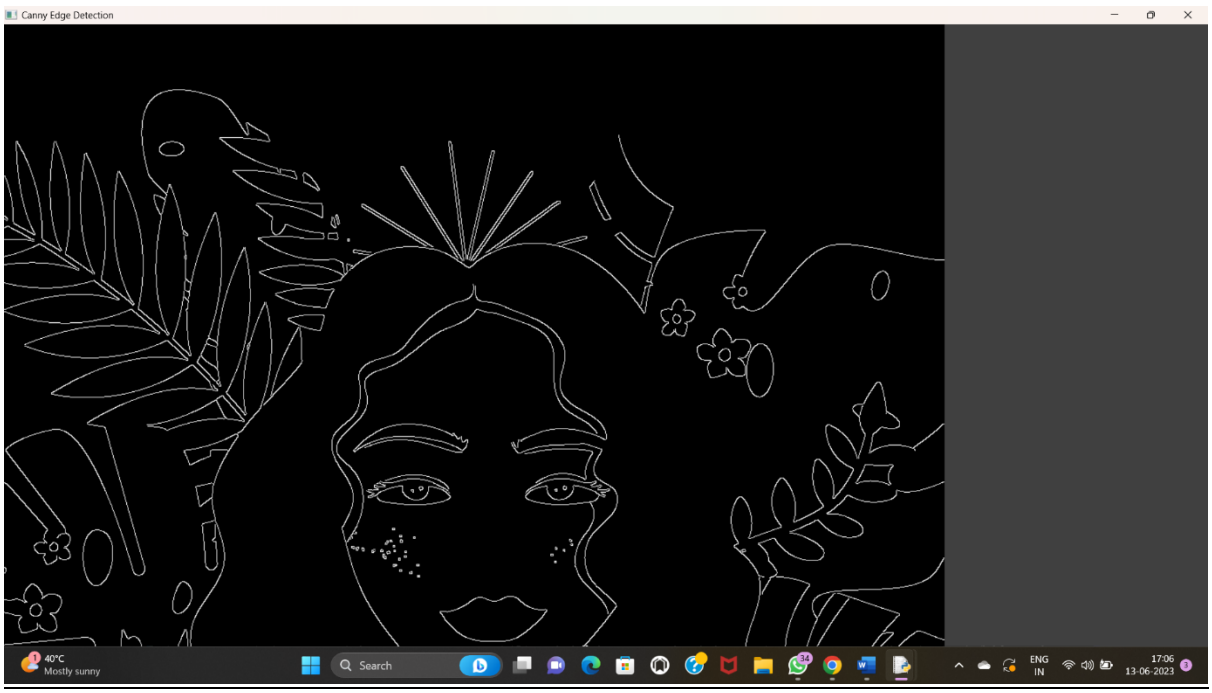


24. Perform Sharpening of Image using High-Boost Masks.

PROGRAM:

```
h_img, w_img, _ = resized_img.shape
center_y = int(h_img/2)
center_x = int(w_img/2)
h_wm, w_wm, _ = resized_wm.shape
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 = resized_img[top_y:bottom_y, left_x:right_x]
result = cv2.addWeighted(roi, 1, resized_wm, 0.3, 0)
resized_img[top_y:bottom_y, left_x:right_x] = result
filename = "C:/Users/divya/Downloads/Girl with a Cat.png"
cv2.imwrite(filename, resized_img)
cv2.imshow("Resized Input Image", resized_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


OUTPUT:

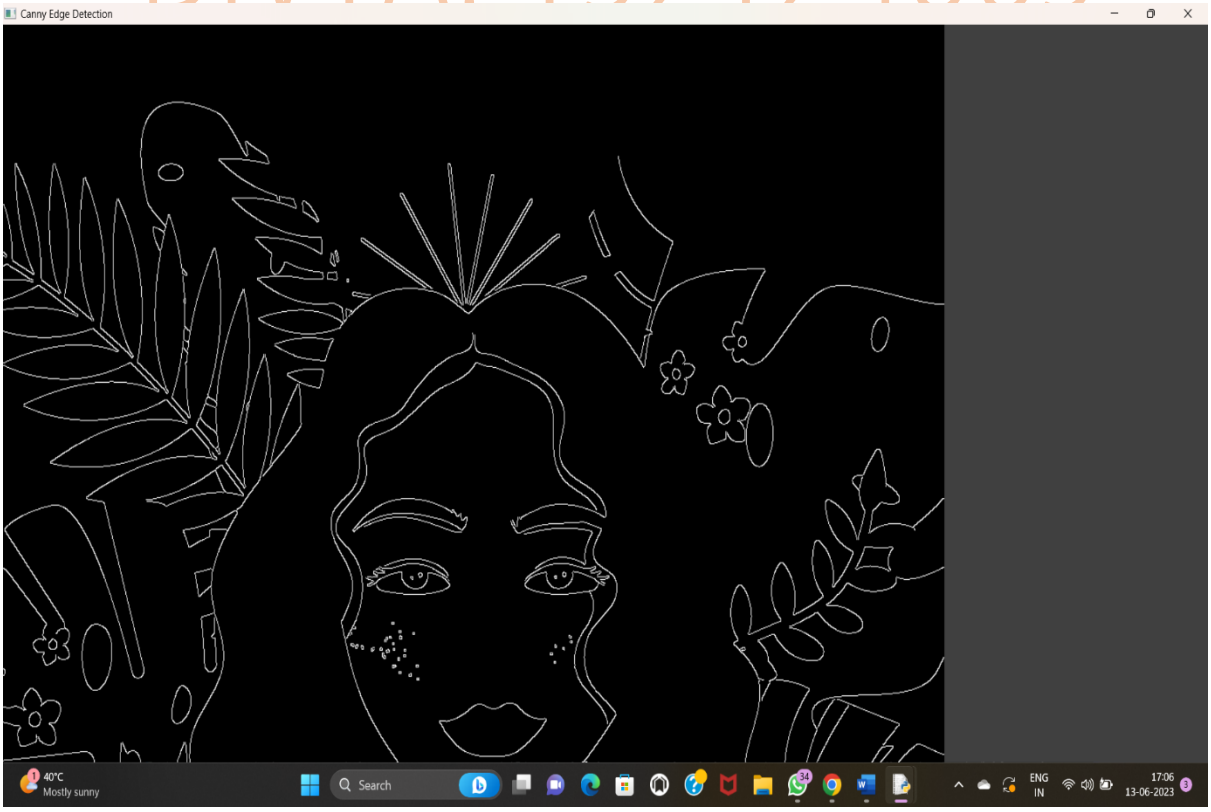


25. Perform Sharpening of Image using Gradient masking

PROGRAM:

```
a=imread("C:/Users/divya/Downloads/Girl with a Cat.png");
Lap=[0, 1, 0, 1,-4, 1, 0, 1, 0];
a1=conv2(a,Lap,"C:/Users/divya/Downloads/Girl with a Cat.png");
a2=uint8(a1);
imtool(abs(a-a2),[])
lap=[-1 ,-1,-1,-1, 8,-1,-1,-1 ,-1];
a3=conv2(a,lap,"C:/Users/divya/Downloads/Girl with a Cat.png");
a4=uint8(a3);
imtool(abs(a+a4),[])
```

OUTPUT:

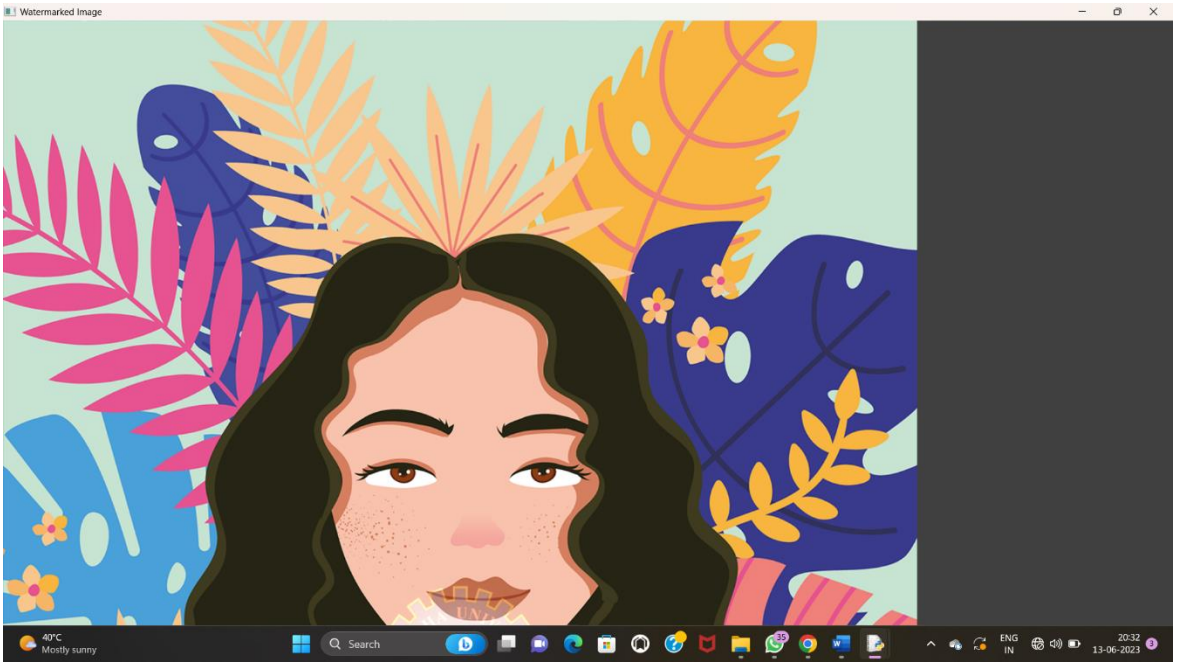


26. Insert water marking to the image using OpenCV.

PROGRAM:

```
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()
```


OUTPUT:

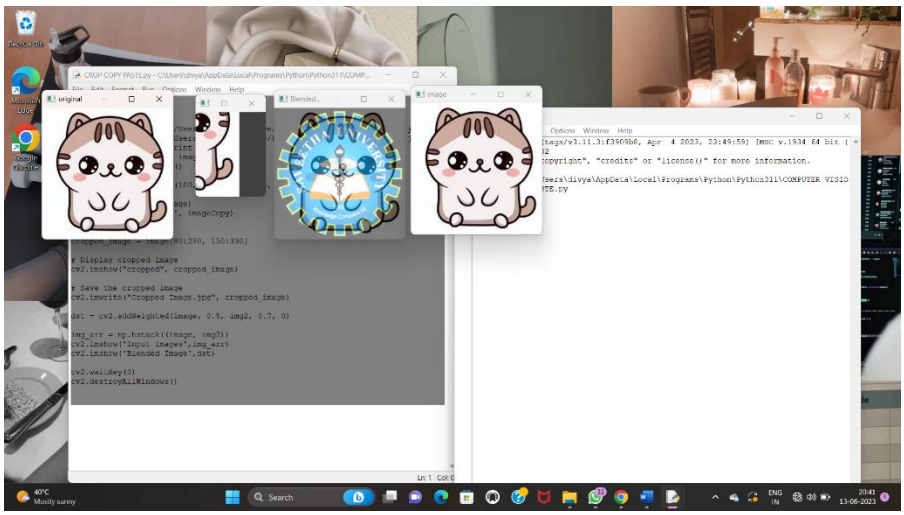


27. Do Cropping, Copying and pasting image inside another image using OpenCV

PROGRAM:

```
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()
```


OUTPUT:

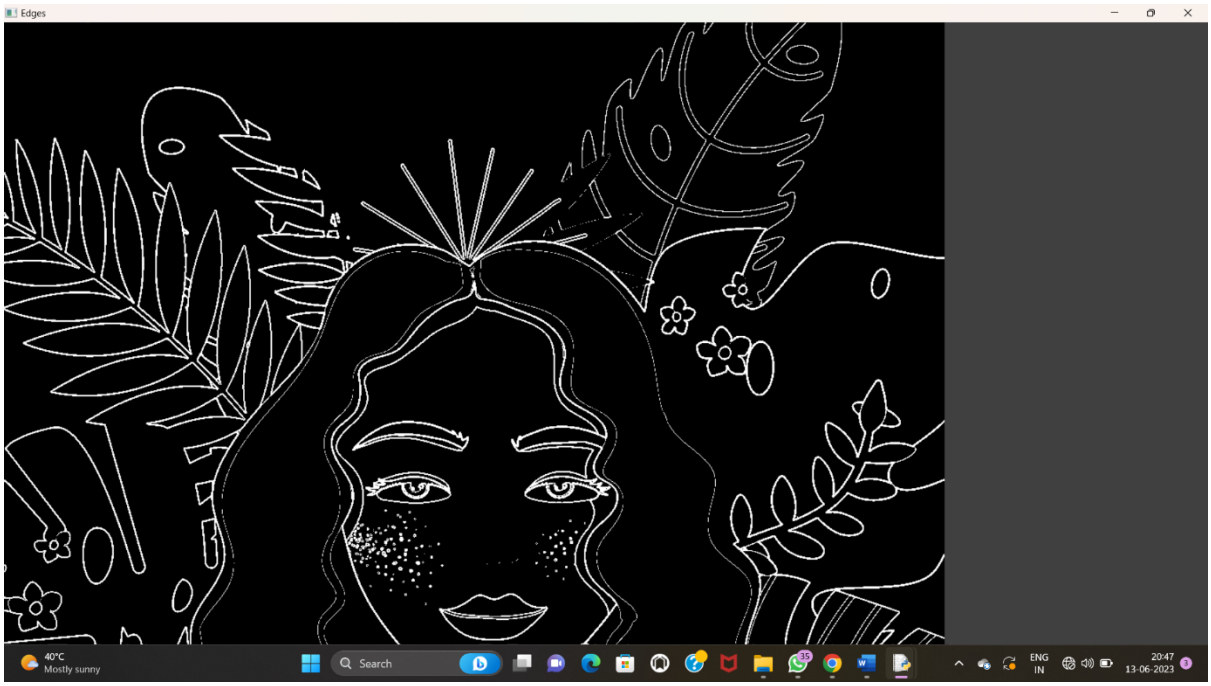


28. Find the boundary of the image using Convolution kernel for the given image

PROGRAM:

```
import cv2

import numpy as np

img = cv2.imread("C:/Users/divya/OneDrive/Documents/COMPUTER VISION/Girl with a Cat.png", cv2.IMREAD_GRAYSCALE)

dx = cv2.Sobel(img, cv2.CV_64F, 1, 0)

dy = cv2.Sobel(img, cv2.CV_64F, 0, 1)

edges = cv2.magnitude(dx, dy)

thresh = 100

edges[edges < thresh] = 0

edges[edges >= thresh] = 255

cv2.imshow("Edges", edges)

cv2.waitKey(0)

cv2.destroyAllWindows()
```


OUTPUT:

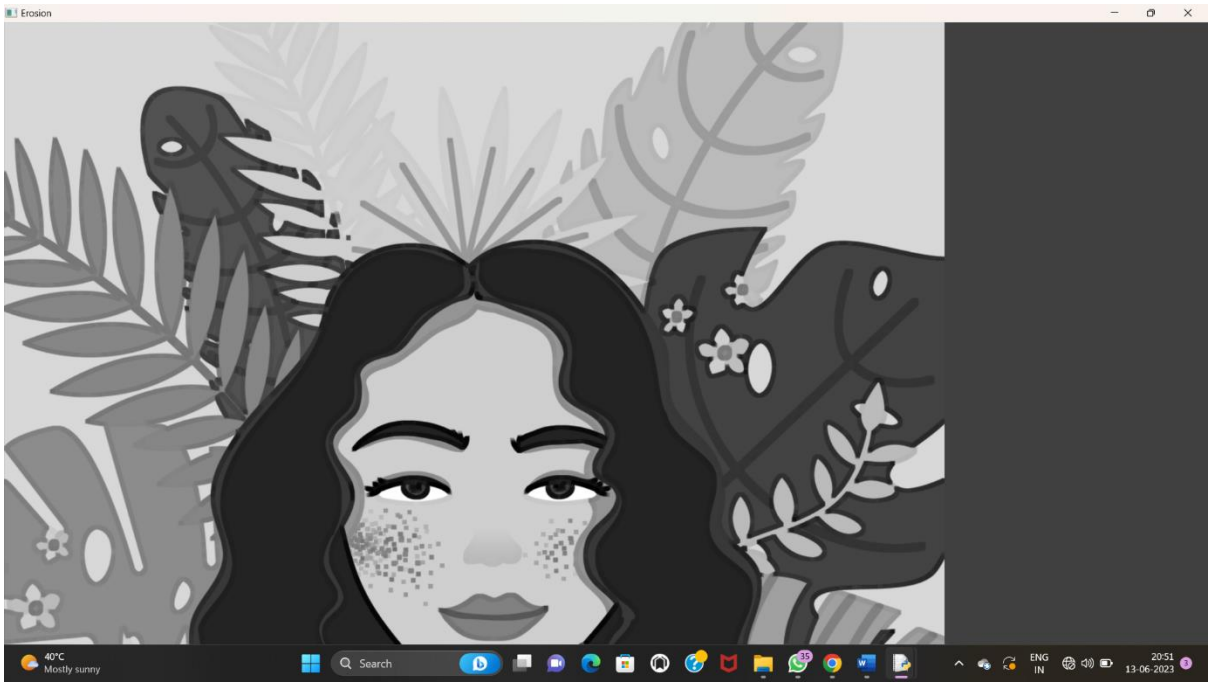


29. Morphological operations based on OpenCV using Erosion technique

PROGRAM:

```
import cv2
import numpy as np
img = cv2.imread("C:/Users/divya/Downloads/Girl with a Cat.png", cv2.IMREAD_GRAYSCALE)
kernel = np.ones((5,5), np.uint8)
erosion = cv2.erode(img, kernel, iterations=1)
cv2.imshow("Original", img)
cv2.imshow("Erosion", erosion)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:

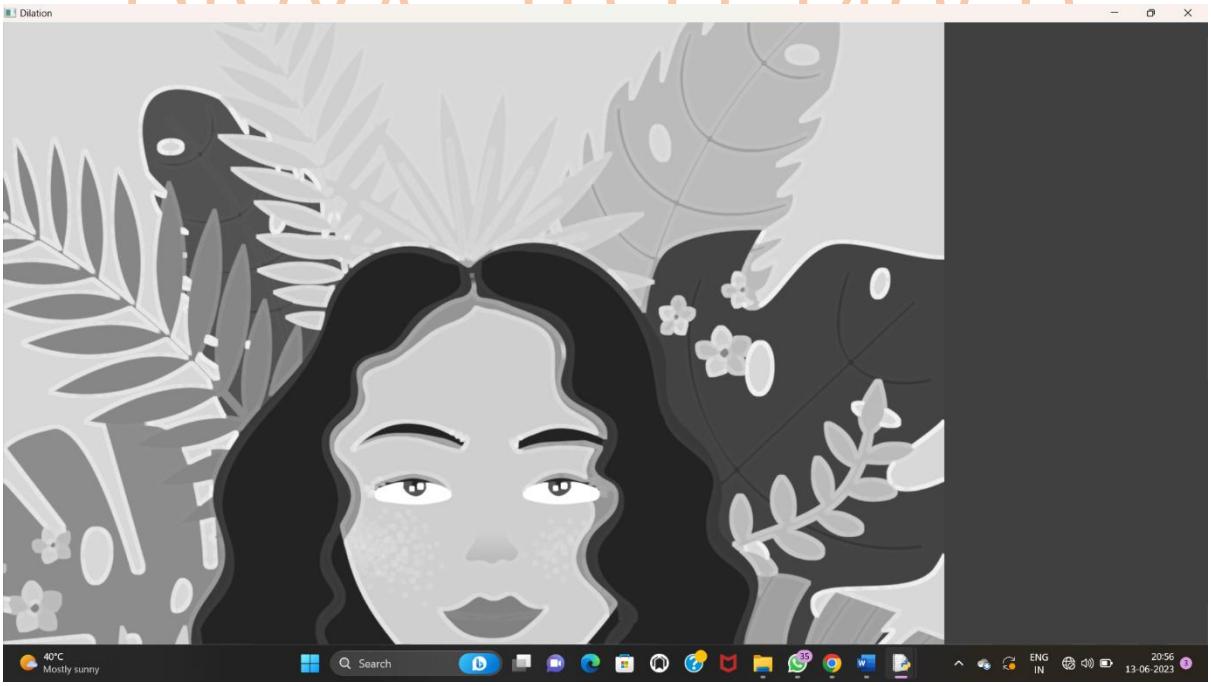


30. Morphological operations based on OpenCV using Dilation technique

PROGRAM:

```
import cv2
import numpy as np
img = cv2.imread("C:/Users/divya/Downloads/Girl with a Cat.png", cv2.IMREAD_GRAYSCALE)
kernel = np.ones((5,5), np.uint8)
dilation = cv2.dilate(img, kernel, iterations=1)
cv2.imshow("Original", img)
cv2.imshow("Dilation", dilation)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:

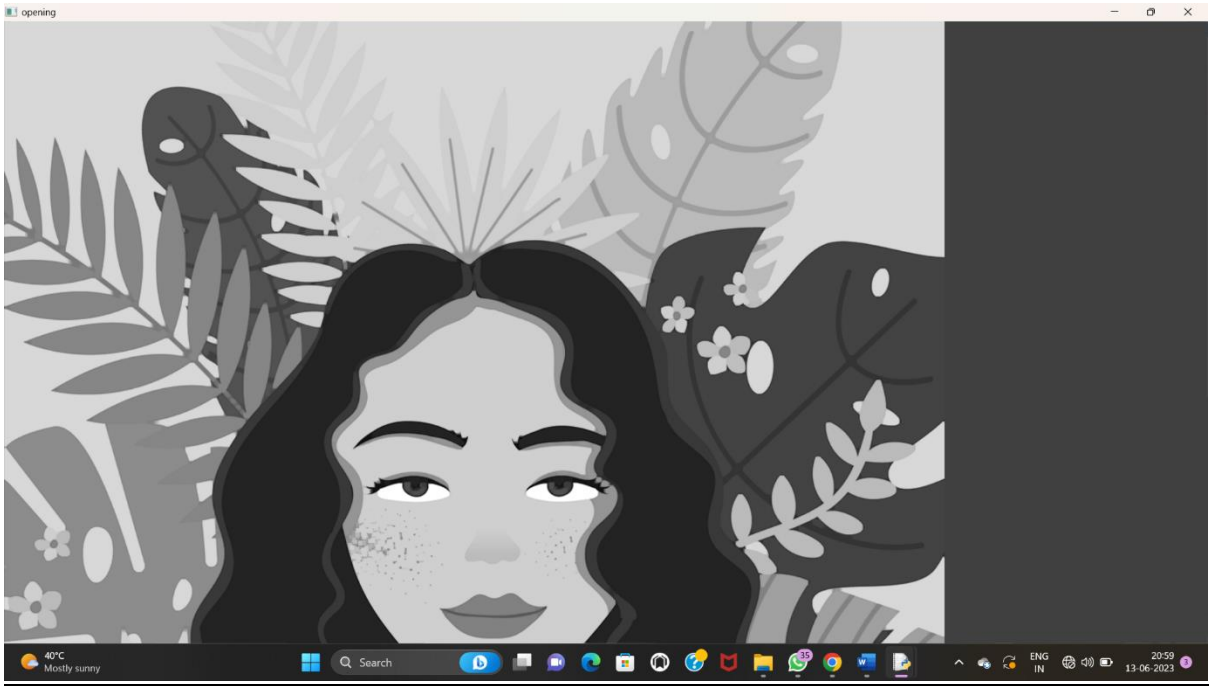


31. Morphological operations based on OpenCV using Opening technique.

PROGRAM:

```
import cv2
import numpy as np
img = cv2.imread("C:/Users/divya/Downloads/Girl with a Cat.png", cv2.IMREAD_GRAYSCALE)
kernel = np.ones((5,5), np.uint8)
opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel)
cv2.imshow("Original", img)
cv2.imshow("opening", opening)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:

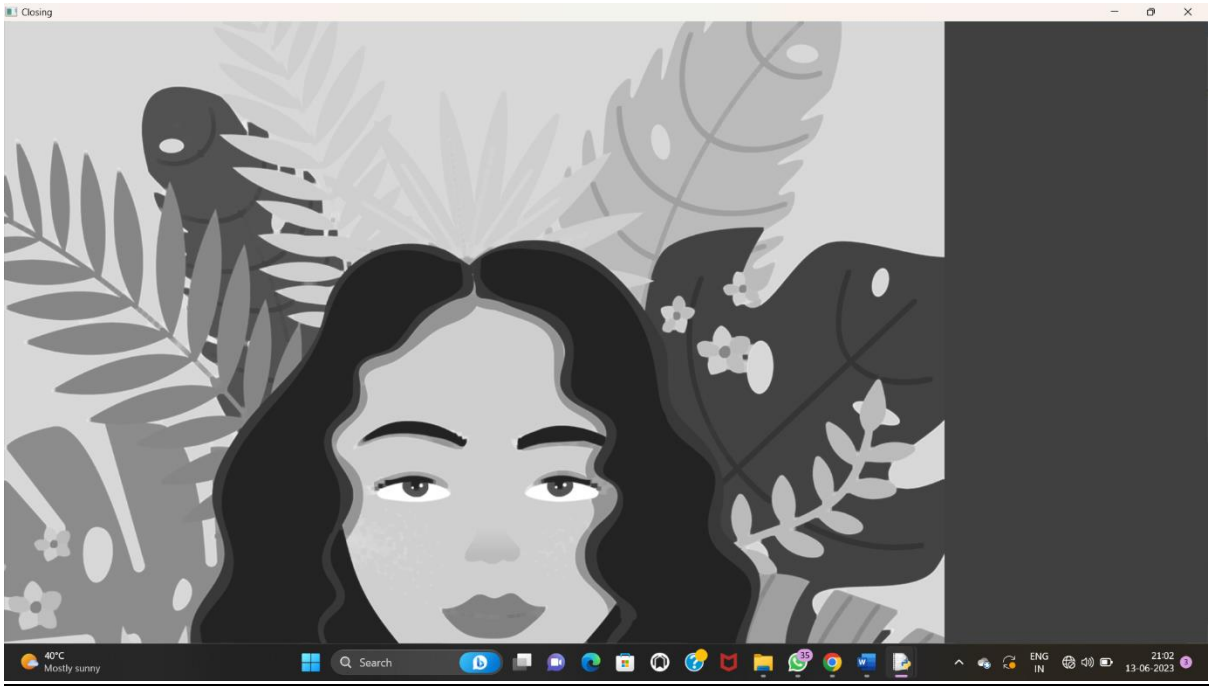


32. Morphological operations based on OpenCV using Closing technique.

PROGRAM:

```
import cv2
import numpy as np
img = cv2.imread("C:/Users/divya/Downloads/Girl with a Cat.png", 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()
```

OUTPUT:

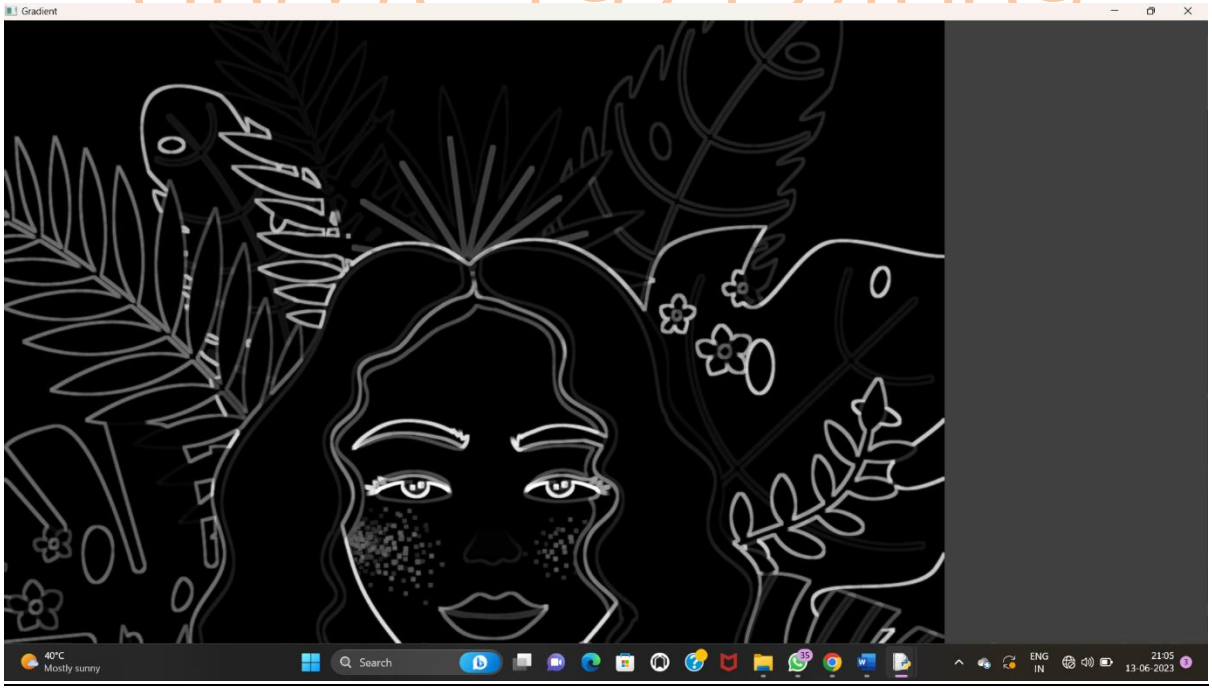


33. Morphological operations based on OpenCV using Morphological Gradient technique

PROGRAM:

```
import cv2

import numpy as np

img = cv2.imread("C:/Users/divya/OneDrive/Documents/COMPUTER VISION/Girl with a Cat.png", cv2.IMREAD_GRAYSCALE)

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

grad = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel)

cv2.imshow("Original", img)

cv2.imshow("Gradient", grad)

cv2.waitKey
```

OUTPUT:

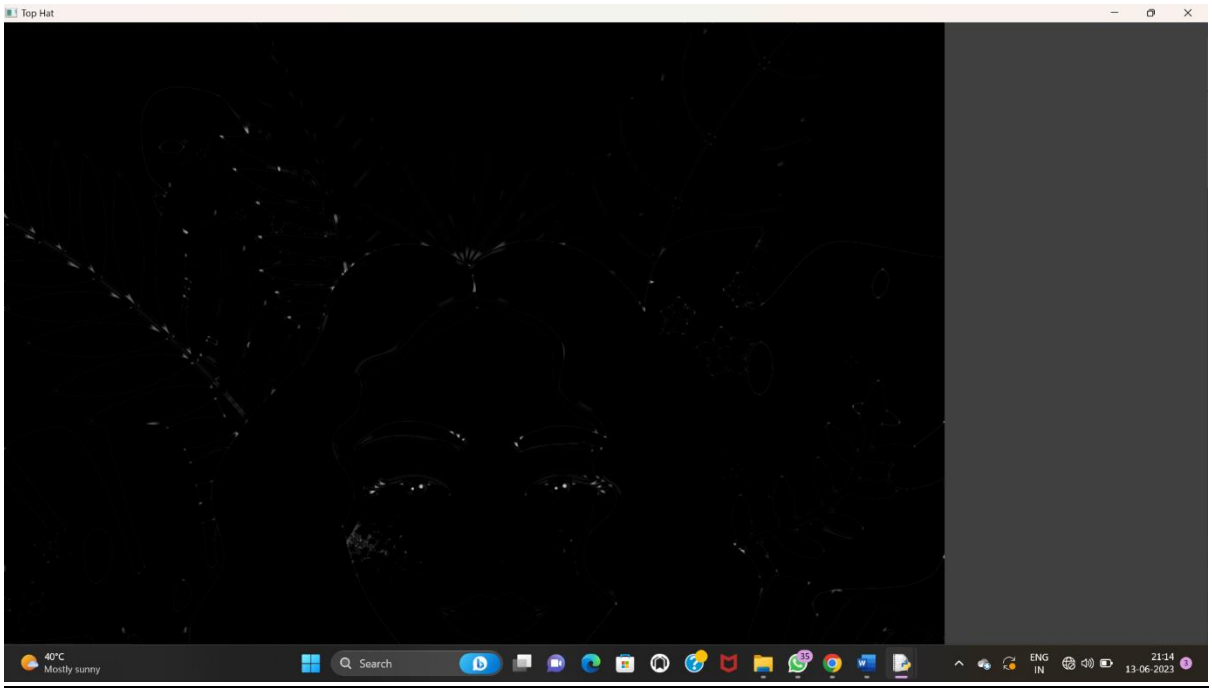


34. Morphological operations based on OpenCV using Top hat technique.

PROGRAM:

```
import cv2

import numpy as np

img = cv2.imread("C:/Users/koppo/Downloads/Genshin-Impact_Key-Art-EN-920x518.png",
cv2.IMREAD_GRAYSCALE)

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

tophat = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel)

cv2.imshow("Original", img)

cv2.imshow("Top Hat", tophat)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:

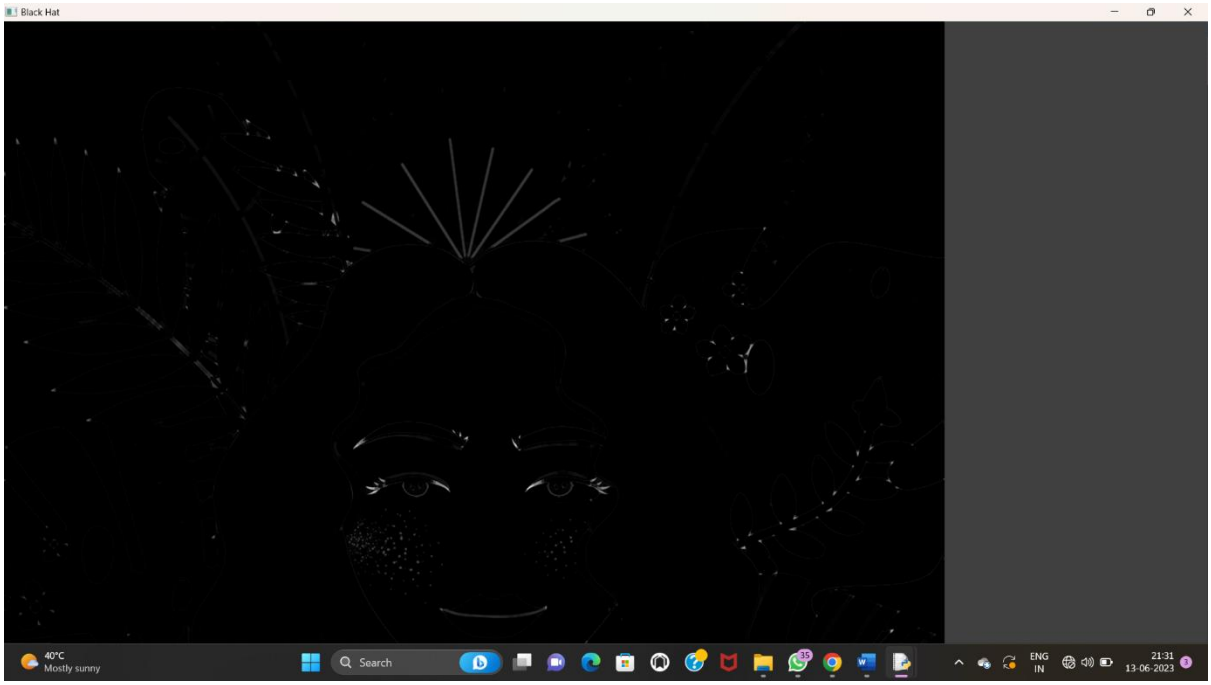


35. Morphological operations based on OpenCV using Black hat technique.

PROGRAM:

```
import cv2

import numpy as np

img = cv2.imread("C:/Users/divya/OneDrive/Documents/COMPUTER VISION/Girl with a Cat.png", cv2.IMREAD_GRAYSCALE)

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

blackhat = cv2.morphologyEx(img, cv2.MORPH_BLACKHAT, kernel)

cv2.imshow("Original", img)

cv2.imshow("Black Hat", blackhat)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:

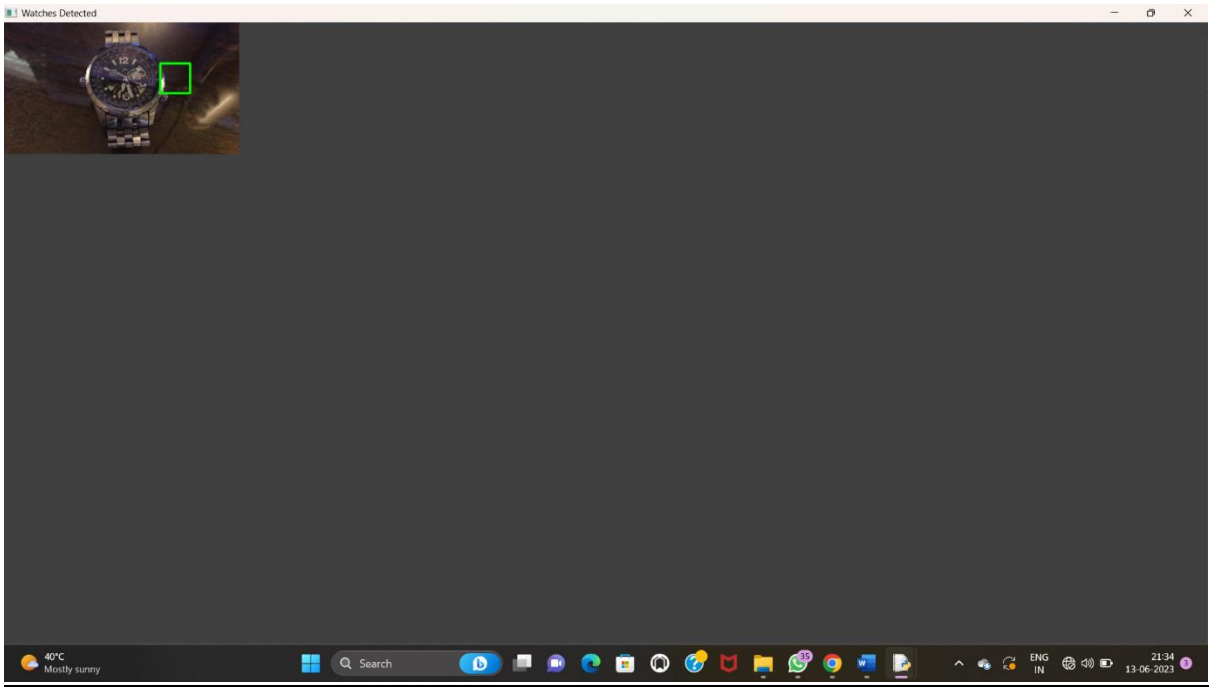


36. Recognise watch from the given image by general Object recognition using OpenCV.

PROGRAM:

```
import cv2

watch_cascade = cv2.CascadeClassifier("C:/Users/divya/OneDrive/Documents/COMPUTER VISION/watch-cascade.xml")

img = cv2.imread("C:/Users/divya/OneDrive/Documents/COMPUTER VISION/COMPUTER VISION/watch.jpg")

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

watches = watch_cascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5)

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

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

cv2.imshow('Watches Detected', img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:

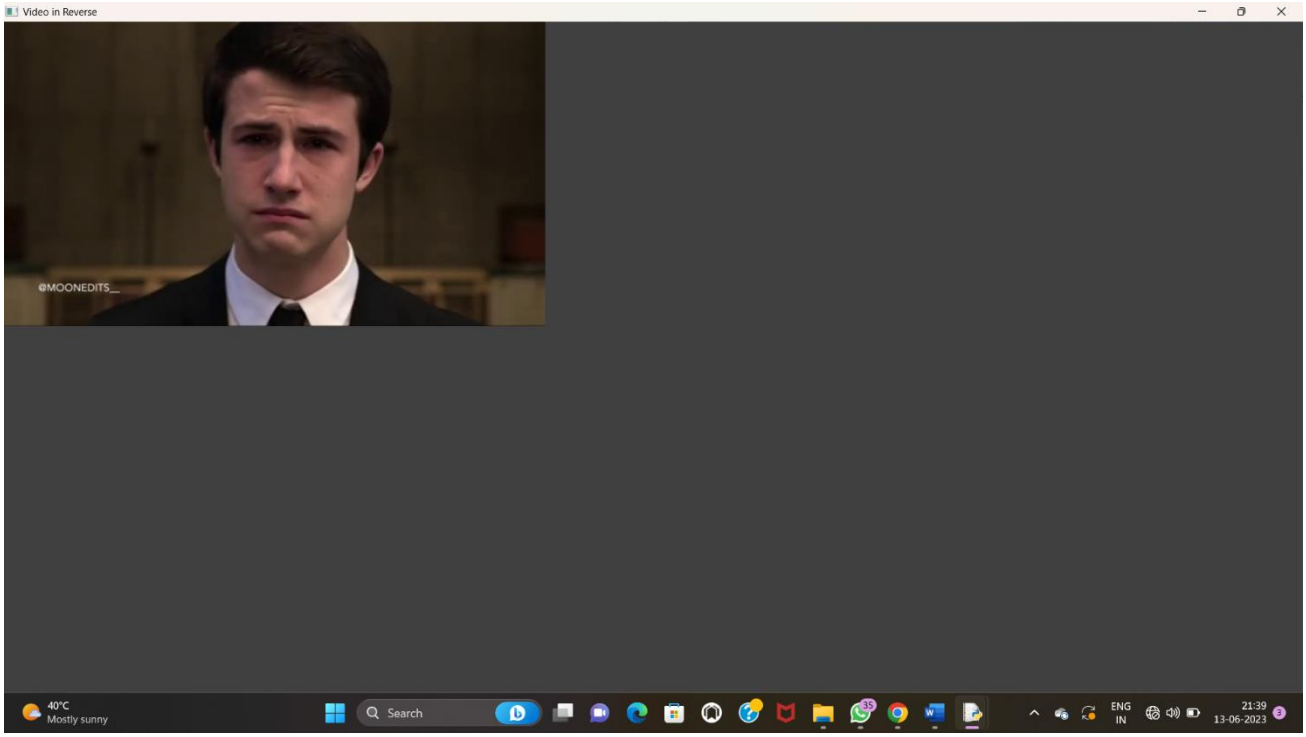


37. Using Opencv play Video in Reverse mode.

PROGRAM:

```
import cv2

cap = cv2.VideoCapture("C:/Users/divya/OneDrive/Documents/COMPUTER VISION/13 REASONS WHY")

total_frames = cap.get(cv2.CAP_PROP_FRAME_COUNT)

current_frame = total_frames- 1

while current_frame >= 0:

    cap.set(cv2.CAP_PROP_POS_FRAMES, current_frame)

    ret, frame = cap.read()

    if not ret:

        break

    cv2.imshow('Video in Reverse', frame)

    if cv2.waitKey(25) & 0xFF == ord('q'):

        break

    current_frame-= 1

cap.release()

cv2.destroyAllWindows()
```


OUTPUT:



38. Face Detection using Opencv

PROGRAM:

```
import cv2

img = cv2.imread("C:/Users/koppo/Downloads/20101123131216-1_0.jpg")

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

face_cascade =
cv2.CascadeClassifier("C:/Users/koppo/Downloads/haarcascade_frontalface_default.xml")

faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

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

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

cv2.imshow('Faces Detected', img)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

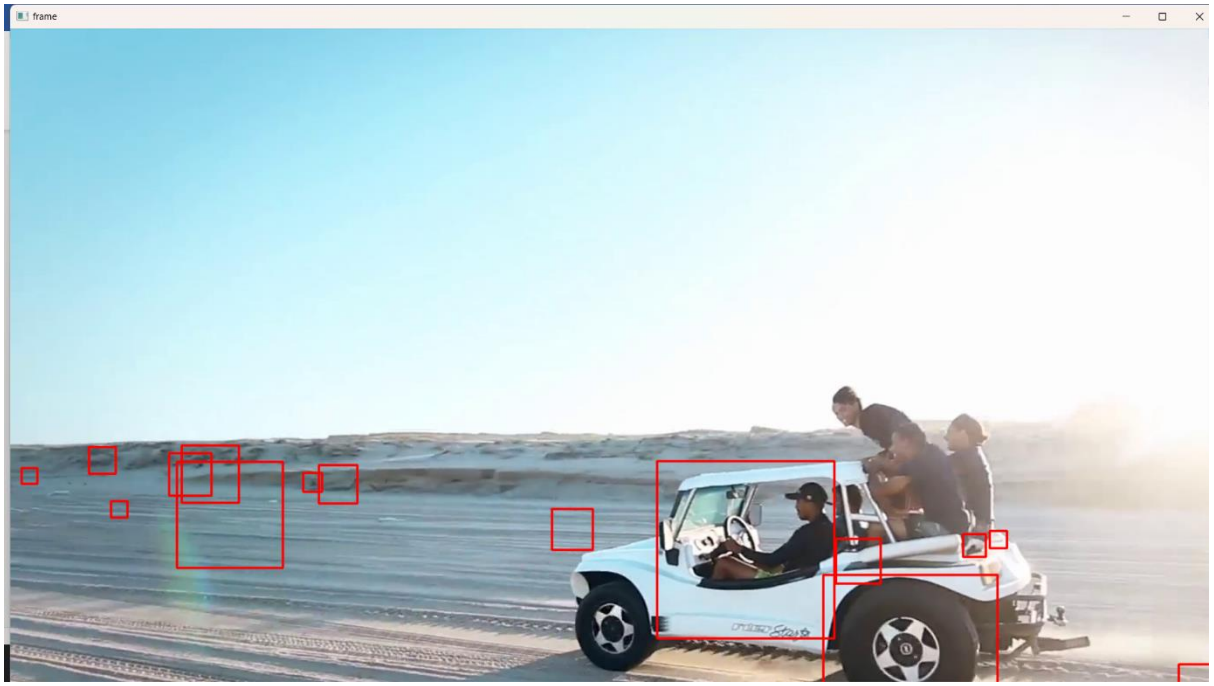
OUTPUT:

39. Vehicle Detection in a Video frame using OpenCV

PROGRAM:

```
import cv2
car_cascade = cv2.CascadeClassifier("C:/Users/divya/OneDrive/Documents/COMPUTER VISION/cars.xml")
cap = cv2.VideoCapture("C:/Users/divya/Downloads/car.mp4")
while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    cars = car_cascade.detectMultiScale(gray, 1.1, 1)
    for (x,y,w,h) in cars:
        cv2.rectangle(frame, (x,y), (x+w,y+h), (0,0,255), 2)
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```


OUTPUT:



40. Draw Rectangular shape and extract objects

PROGRAM:

```
import cv2
img = cv2.imread("C:/Users/divya/OneDrive/Documents/COMPUTER VISION/40.jpg")
x, y = 100, 100
width, height = 200, 150
roi = img[y:y+height, x:x+width]
cv2.imshow('ROI', roi)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT: