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

Program:

import cv2

import numpy as np

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

print(kernel)

path = "C:/Users/Rakesh/Pictures/Screenshots/Screenshot (68).png"

img =cv2.imread(path)

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

cv2.imshow("GrayScale",imgGray)

cv2.waitKey(0)

input:



Output:



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

Program:

import cv2

import numpy as np

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

print(kernel)

path="C:/Users/Rakesh/Pictures/Screenshots/Screenshot (68).png"

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)

output



1. 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

Program:

import cv2

import numpy as np

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

print(kernel)

path = "C:/Users/Rakesh/Pictures/Screenshots/Screenshot (68).png"

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)



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

PROGRAM:

import cv2

import numpy as np

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

print(kernel)

path ="C:/Users/Rakesh/Pictures/air.png"

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)



1. Perform basic Image Handling and processing operations on the image• 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/Rakesh/Pictures/air.png"

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)



1. 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.

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()



1. 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)

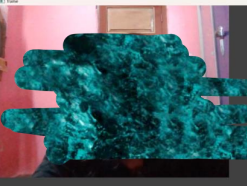
if k == ord("q"):

break

cap.release()

output.release()

cv2.destroyAllWindows()



1. Scaling an image to its Bigger and Smaller sizes

Program:

import cv2

import numpy as np

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

img = cv2.imread("C:/Users/Rakesh/Pictures/air.png")

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

cv2.imshow("image",img)

cv2.waitKey(0)



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

Program:

ROTATION 90 ALONG DEGREE:

import cv2

path ("C:/Users/Rakesh/Pictures/air.png")

src = cv2.imread(path)

window\_name = 'Image'

image = cv2.rotate(src, cv2.ROTATE\_180)

cv2.imshow(window\_name, image)

cv2.waitKey(0)



ROTATION ALONG 180 DEGRE

import cv2

path = "C:/Users/Rakesh/Pictures/air.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)



ROTATION ALONG 270 DEGREE

import cv2

path = "C:/Users/Rakesh/Pictures/air.png"

src = cv2.imread(path)

window\_name = 'Image'

image = cv2.rotate(src, cv2.ROTATE\_90\_COUNTERCLOCKWISE)

cv2.imshow(window\_name, image)

cv2.waitKey(0)



11.Perform Affine Transformation on the image.

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()



12. Perform Perspective Transformation on the image.

INPUT:

# import required libraries

import cv2

import numpy as np

# read the input image

img = cv2.imread("**D:/Downloads/bLACK.png**")

# 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.

Input:

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()

**OUTPUT:**



14. Perform transformation using Homography matrix.

**Input:**

import cv2

import numpy as np

# Read source image.

im\_src = cv2.imread("**D:/Downloads/bLACK.png**")

# 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("**D:/Downloads/bLACK.png**")

# 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

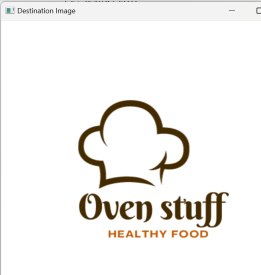
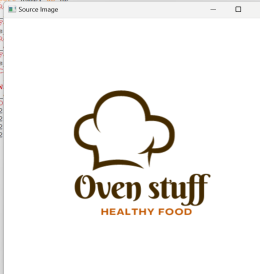
cv2.imshow("Source Image", im\_src)

cv2.imshow("Destination Image", im\_dst)

cv2.imshow("Warped Source Image", im\_out)

cv2.waitKey(0)

OUTPUT:

****

15. Perform transformation using Direct Linear Transformation.

INPUT:

import cv2

import numpy as np

# Load images

img1 = cv2.imread("**D:/Downloads/bLACK.png**")

img2 = cv2.imread("**D:/Downloads/bLACK.png**")

# 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)

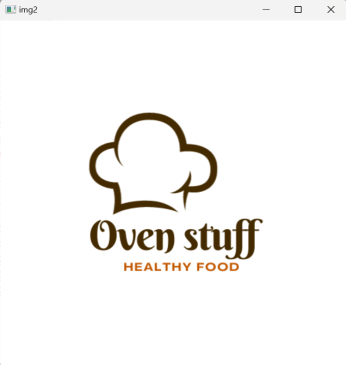
cv2.imshow('img2', img2)

cv2.imshow('dst', dst)

cv2.waitKey(0)

cv2.destroyAllWindows()

**OUTPUT:**

****

16. Perform Edge detection using canny method.

**Input:**

import cv2

img = cv2.imread("D:/Downloads/bLACK.png")

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()

**OUTPUT:**

****

17. Perform Edge detection using Sobel Matrix along X axis.

INPUT:

import cv2

img = cv2.imread("D:/Downloads/bLACK.png")

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)

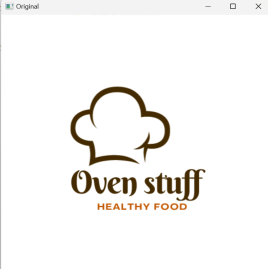
sobelx = cv2.Sobel(src=img\_blur, ddepth=cv2.CV\_64F, dx=1, dy=0, ksize=5) # Sobel Edge Detection on

the X axis

cv2.imshow('Sobel X', sobelx)

cv2.waitKey(0)

**OUTPUT:**

****

****

18. Perform Edge detection using Sobel Matrix along Y axis.

INPUT:

import cv2

# Read the original image

img = cv2.imread("D:/Downloads/bLACK.png")

# 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

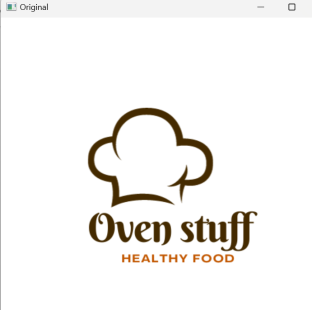
#Detection on the Y axis

# Display Sobel Edge Detection Images

cv2.imshow('Sobel Y', sobely)

cv2.waitKey(0)

OUTPUT:

****

19. Perform Edge detection using Sobel Matrix along XY axis.

Input:

import cv2

img = cv2.imread("D:/Downloads/bLACK.png")

# 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)

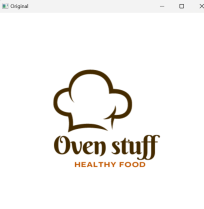
sobelxy = cv2.Sobel(src=img\_blur, ddepth=cv2.CV\_64F, dx=1, dy=1, ksize=5) # Combined X and Y

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.

import cv2

import numpy as np

img = cv2.imread("D:/Downloads/bLACK.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)

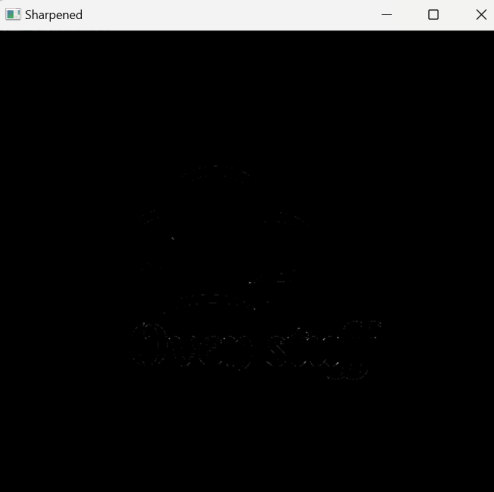
cv2.imshow('Original', gray)

cv2.imshow('Sharpened', sharpened)

cv2.waitKey(0)

cv2.destroyAllWindows()

OUTPUT:

****

21.

import cv2

import numpy as np

img = cv2.imread("C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg")

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()

Op:



22.import cv2

import numpy as np

img = cv2.imread("C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg")

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()

op:



23.import cv2

import numpy as np

img = cv2.imread("C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg")

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)

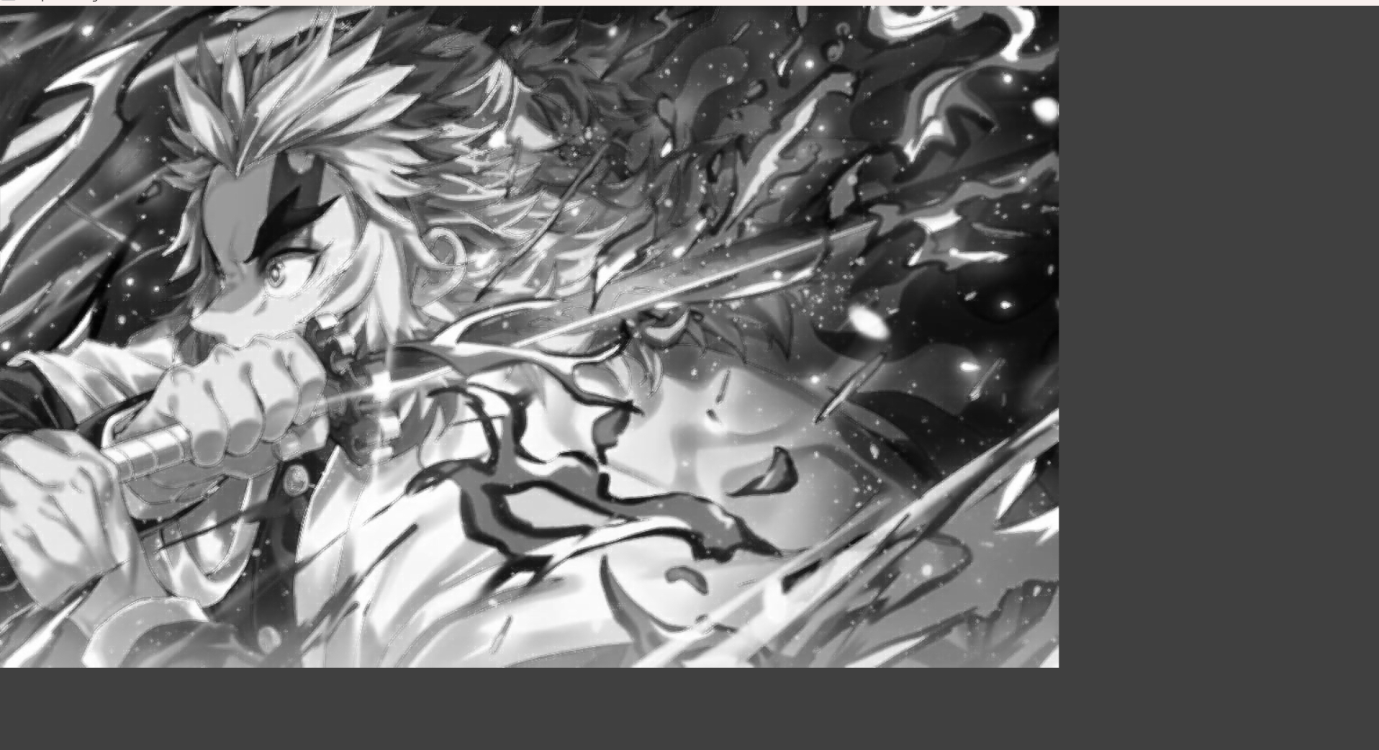
cv2.imshow('Original Image', gray)

cv2.imshow('Sharpened Image', sharpened)

cv2.waitKey(0)

cv2.destroyAllWindows()

op:



24.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/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg"

cv2.imwrite(filename, resized\_img)

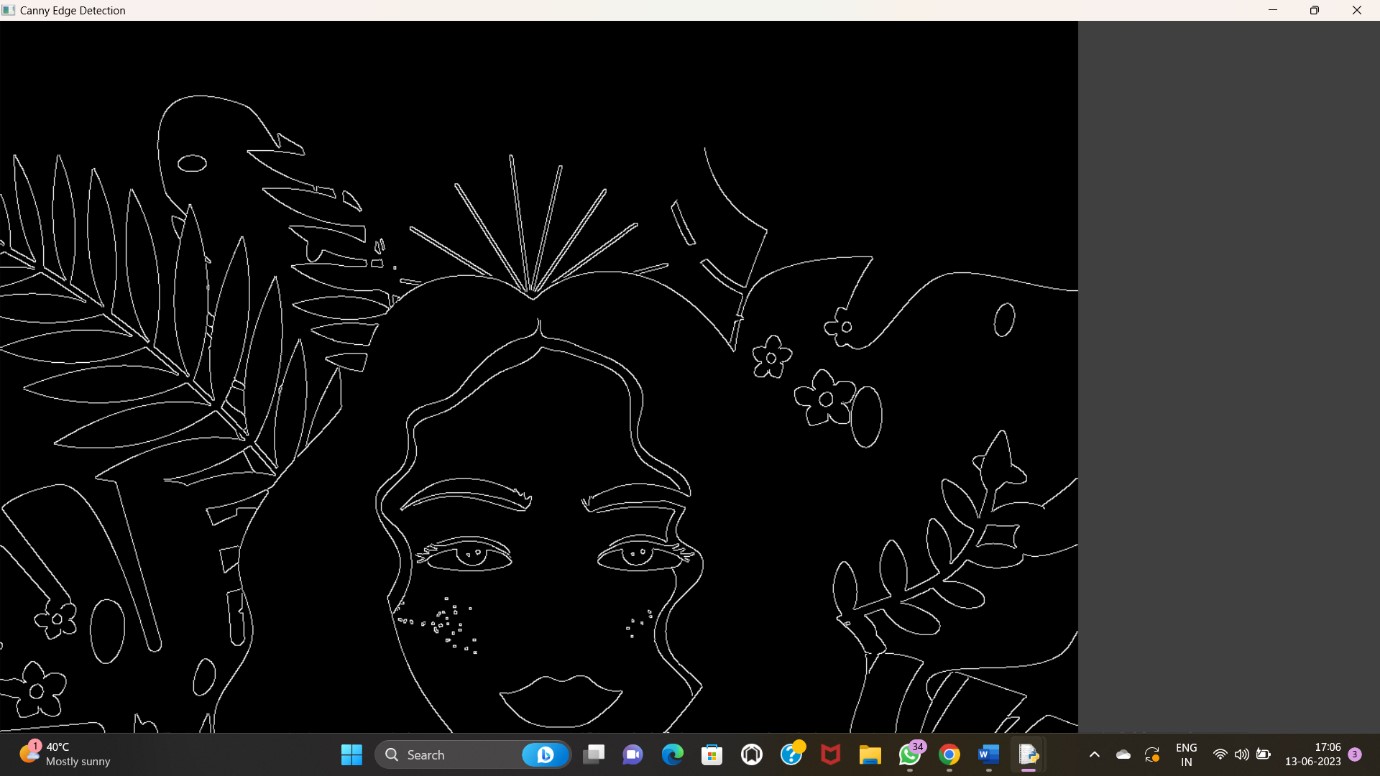
cv2.imshow("Resized Input Image", resized\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

op:

OUTPUT:



25.a=imread("C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg");

Lap=[0, 1, 0, 1, -4, 1, 0, 1, 0];

a1=conv2(a,Lap,"C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg");

a2=uint8(a1); imtool(abs(a-a2),[])

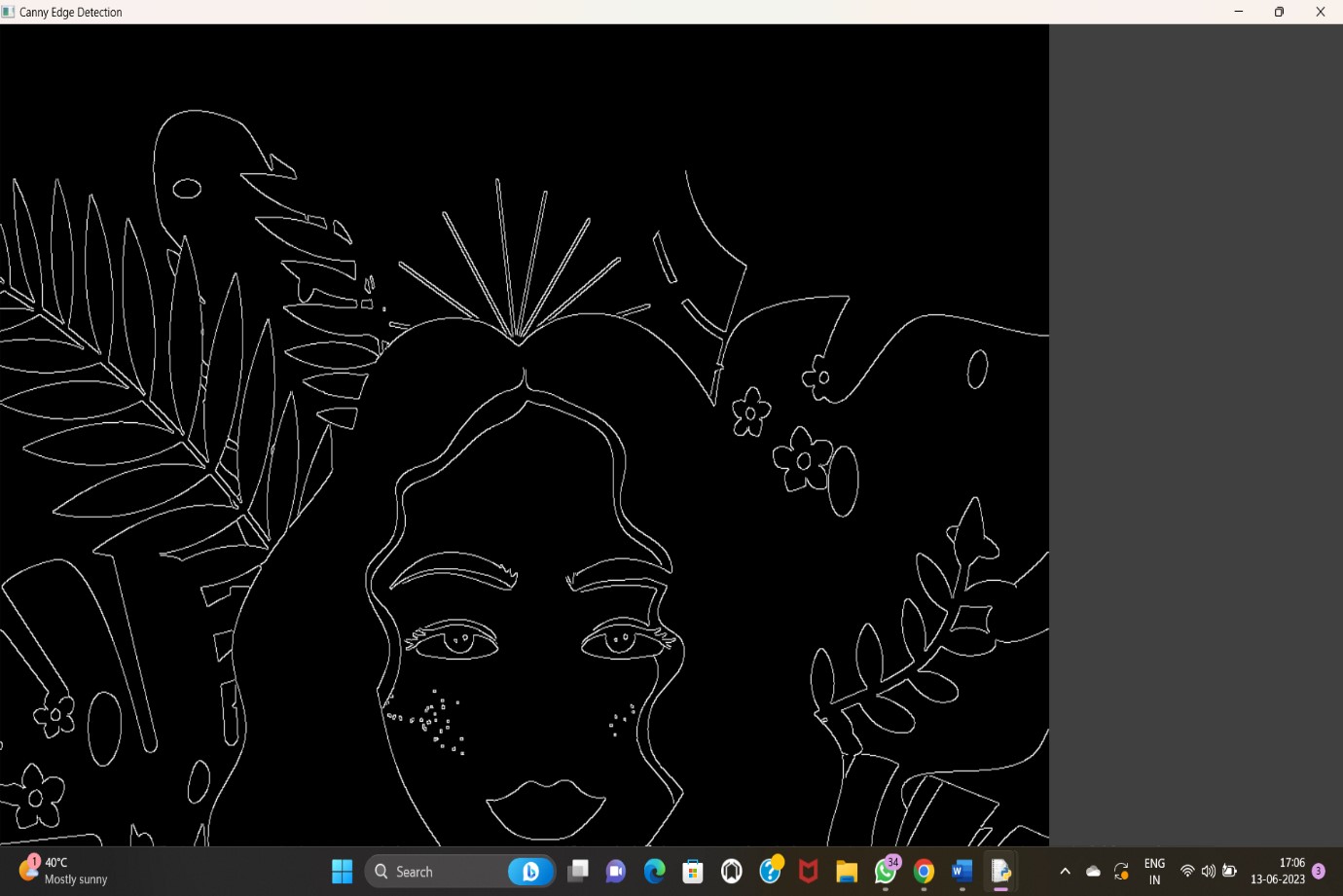
lap=[-1 ,-1, -1, -1, 8, -1, -1, -1 ,-1];

a3=conv2(a,lap,"C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg");

a4=uint8(a3);

imtool(abs(a+a4),[])

OP:



26.import cv2

img = cv2.imread("C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg")

wm = cv2.imread("C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg")

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)

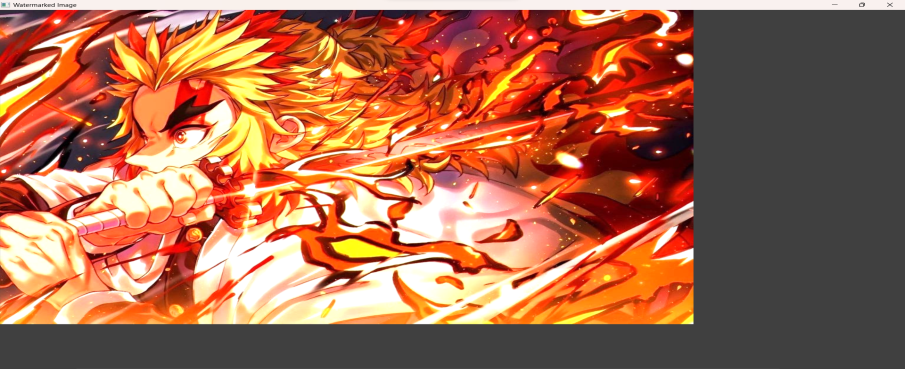
img[top\_y:bottom\_y, left\_x:right\_x] = result

cv2.imshow("Watermarked Image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

Op:



27.import cv2

import numpy as np

image = cv2.imread("C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg")

img2 = cv2.imread("C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg")

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()

Op:



28.import cv2

import numpy as np

img = cv2.imread("C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg", 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

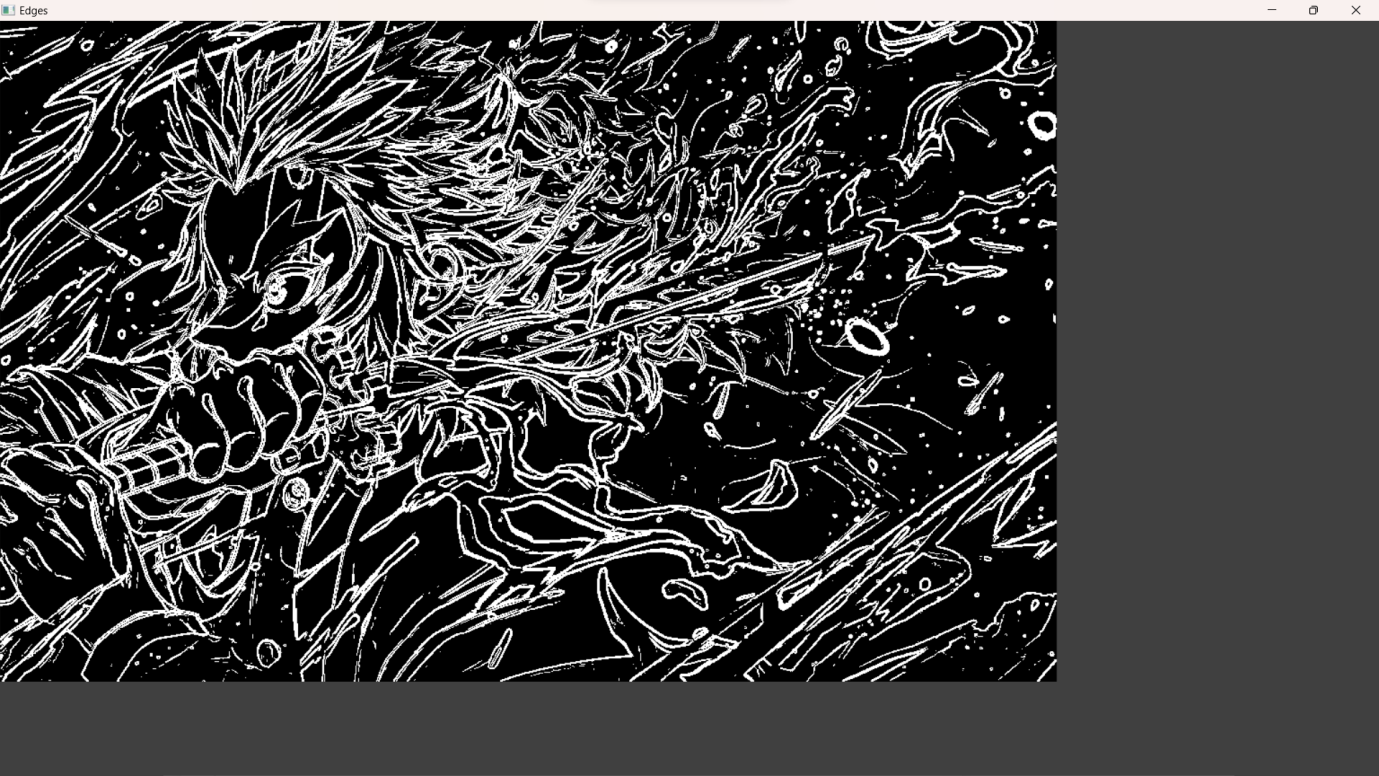
edges[edges >= thresh] = 255

cv2.imshow("Edges", edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

op:



29.import cv2

import numpy as np

img = cv2.imread("C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg", cv2.IMREAD\_GRAYSCALE)

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

erosion = cv2.erode(img, kernel, iterations=1)

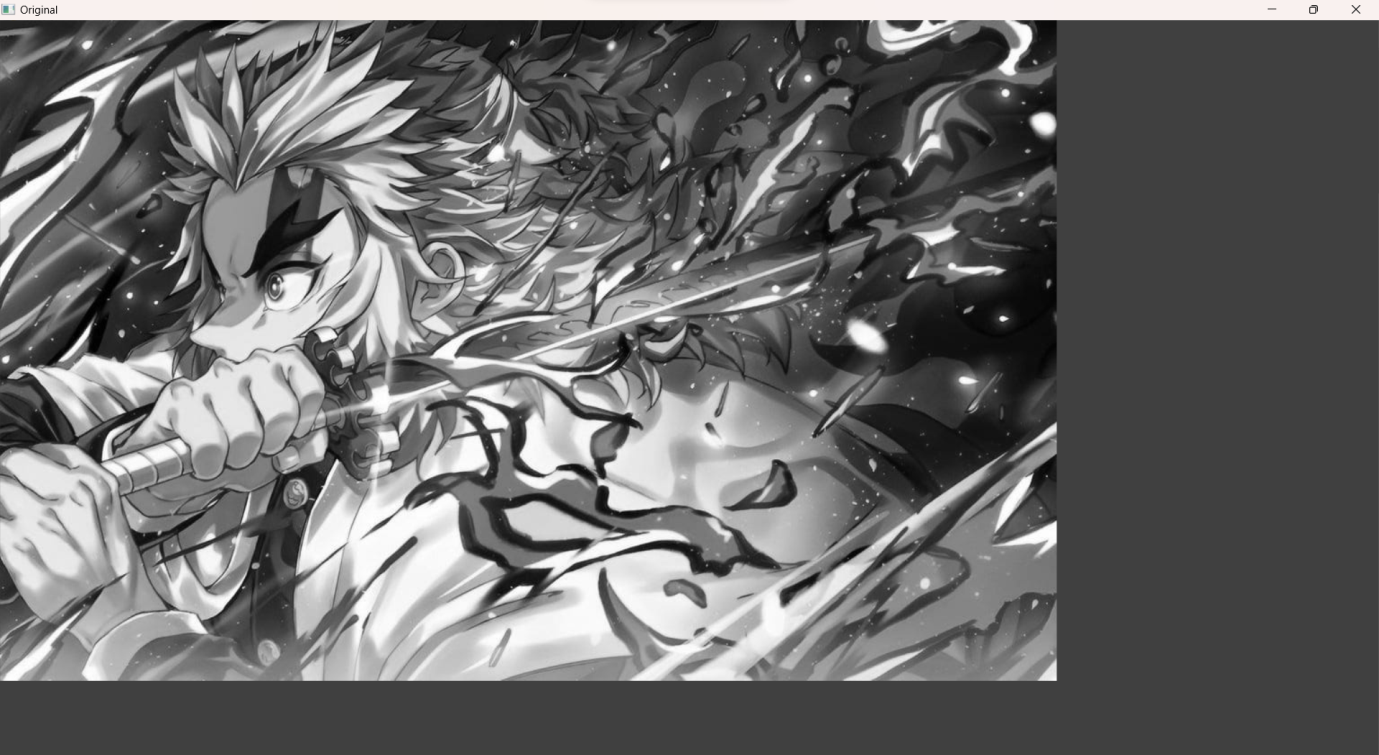
cv2.imshow("Original", img)

cv2.imshow("Erosion", erosion)

cv2.waitKey(0)

cv2.destroyAllWindows()

op:



30.import cv2

import numpy as np

img = cv2.imread "C:/Users/justin shawn allen/OneDrive/Pictures/WhatsApp Image 2023-07-10 at 15.15.09.jpg"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()



31. Morphological operations based on OpenCV using Opening technique.

Program:

import cv2

import numpy as np

img = cv2.imread("C:/Users/Rakesh/Pictures/air.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()





32. Morphological operations based on OpenCV using Closing technique

import cv2

import numpy as np

img = cv2.imread("C:/Users/Rakesh/Pictures/air.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()



33. Morphological operations based on OpenCV using Morphological Gradient technique

Program:

import cv2

import numpy as np

img = cv2.imread("C:/Users/Rakesh/Pictures/air.png", cv2.IMREAD\_GRAYSCALE)

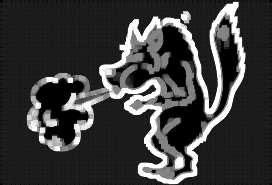
kernel = np.ones((5,5), np.uint8)

grad = cv2.morphologyEx(img, cv2.MORPH\_GRADIENT, kernel)

cv2.imshow("Original", img)

cv2.imshow("Gradient", grad)

cv2.waitKey(0)



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

import cv2

import numpy as np

img = cv2.imread("C:/Users/Rakesh/Pictures/air.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()



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

import cv2

import numpy as np

img = cv2.imread("C:/Users/Rakesh/Pictures/air.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() 

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

import cv2

watch\_cascade = cv2.CascadeClassifier("C:/Users/Rakesh/Pictures/air.png", VISION/watch-cascade.xml)

img = cv2.imread("C:/Users/Rakesh/Pictures/10.jpg",VISION/watch.jpg)

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

watches = watch\_cascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5)

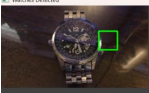
for (x, y, w, h) in watches:

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

cv2.imshow('Watches Detected', img)

cv2.waitKey(0)

cv2.destroyAllWindows()



37. Using Opencv play Video in Reverse mode.

import cv2

cap = cv2.VideoCapture("C:/Users/Rakesh/Downloads/mylivewallpapers-com-Neon-Glow-Demon-Slayer.mp4")

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()



38. Face Detection using Openc

import cv2

img = cv2.imread("C:/Users/Rakesh/Pictures/air.png")

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

face\_cascade = cv2.CascadeClassifier("C:/Users/Rakesh/Pictures/air.png")

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()

39. Vehicle Detection in a Video frame using OpenCV

import cv2

car\_cascade = cv2.CascadeClassifier("C:/Users/OneDrive/Documents/COMPUTER VISION/cars.xml")

cap = cv2.VideoCapture("C:/Users/Rakesh/Pictures/air.png")

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)

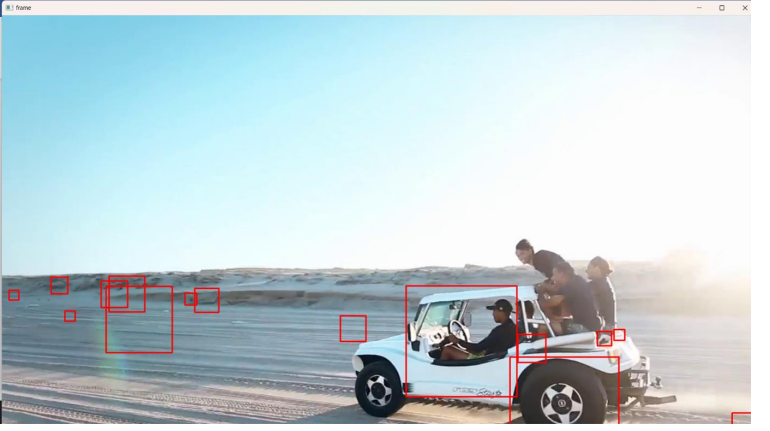
cv2.imshow('frame', frame)

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

break

cap.release()

cv2.destroyAllWindows()



40. Draw Rectangular shape and extract objects

import cv2

img = cv2.imread("C:/Users/Rakesh/Pictures/air.png")

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()

