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

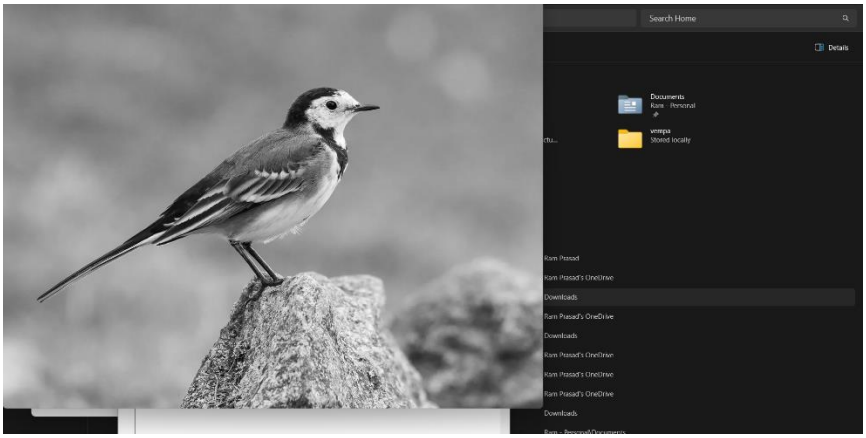
AIM: To Perform Basic Operations to Read Image and Convert to Grayscale using Python .

Program :

```
import cv2
import numpy as np
kernel = np.ones((5,5),np.uint8)
print(kernel)
path ="C:\drive\OneDrive\Pictures\pass photo.jpg"
img =cv2.imread(path)
imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
cv2.imshow("GrayScale",imgGray)
cv2.waitKey(0)
```

INPUT :

OUTPUT:



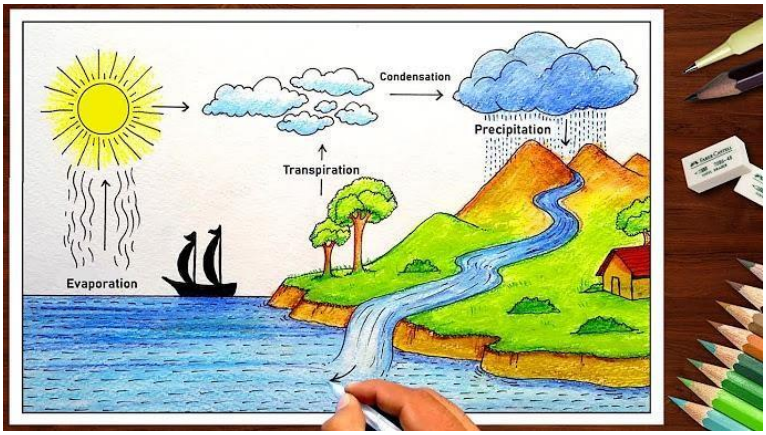
2. Perform basic Image Handling and processing operations on the image.• Read an image in python and Convert an Image to Blur using GaussianBlur.

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

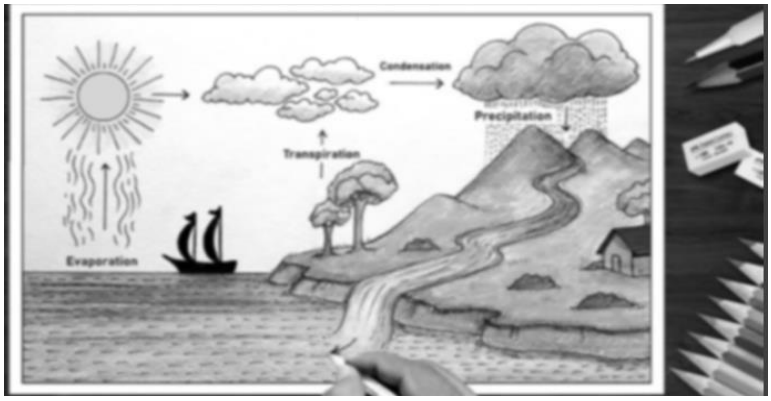
PROGRAM :

```
import cv2
import numpy as np
kernel = np.ones((5,5),np.uint8)
print(kernel)
path = "C:/Users/vempa/Downloads/lab 2.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 :



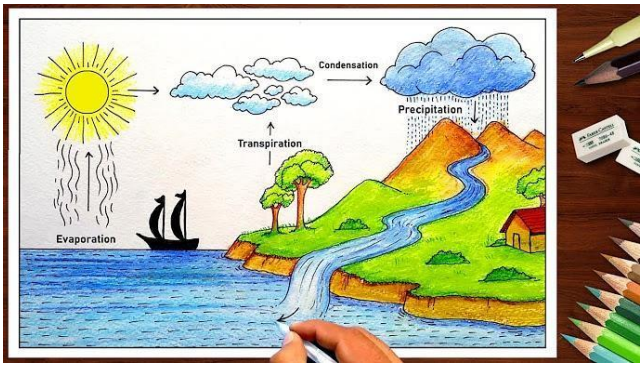
3. Perform basic Image Handling and processing operations on the image• Read an image in python and Convert an Image to show outline using Canny function

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

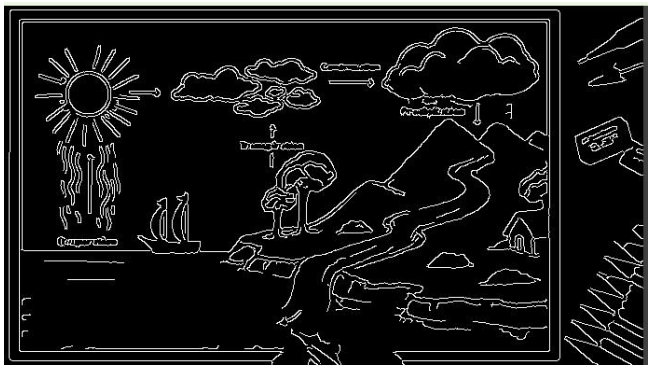
PROGRAM:

- import cv2
- import numpy as np
- kernel = np.ones((5,5),np.uint8)
- print(kernel)
- path = "C:/Users/vempa/Downloads/lab 2.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

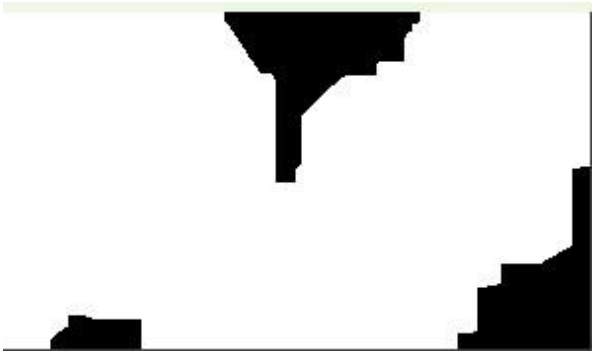
AIM: To Perform Basic Operations to Read Image and Dilate an Image using Python

PROGRAM:

```
import cv2
import numpy as np
kernel = np.ones((5,5),np.uint8)
print(kernel)
path = "C:/Users/vempa/Downloads/LAB4.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

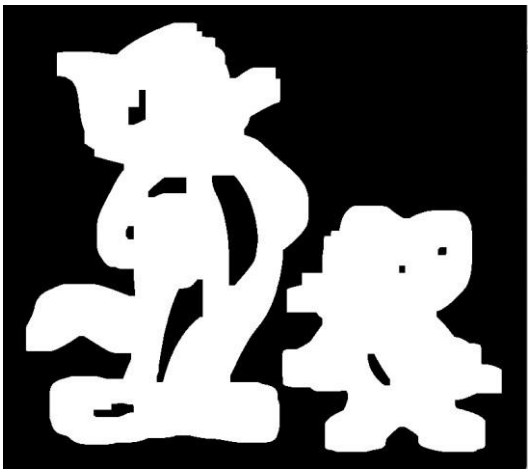
AIM: The Aim of the experiment is to Read an image in python and Erode an Image using erode function

PROGRAM:

```
import cv2
import numpy as np
kernel = np.ones((5,5),np.uint8)
print(kernel)
path ="C:/Users/vempa/Downloads/HD-wallpaper-tom-and-jerry-cartoons.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)
```

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 def play_video(video_path,
speed=1.0):     cap =
cv2.VideoCapture(video_path)     if not
cap.isOpened():
      print("Error opening video file")
return
   fps = cap.get(cv2.CAP_PROP_FPS)
new_fps = fps * speed     while
cap.isOpened():        ret, frame =
cap.read()        if not ret:
break
      cv2.imshow('Video Player', frame)
      if cv2.waitKey(int(1000 / new_fps)) & 0xFF == 27:  # Press 'Esc' to exit
         break
cap.release()
cv2.destroyAllWindows()
video_path = "C:/drive/OneDrive/Pictures/Slide Shows/Ram's/WA-VID-20200720-9aa8edb7.mp4"
play_video(video_path, speed=0.5) play_video(video_path, speed=2.0)
```

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 def display_video_slow_fast(video_path, slow_factor=0.5, fast_factor=2.0):
   cap = cv2.VideoCapture(video_path)     if not cap.isOpened():
      print("Error: Could not open video device or file.")        return
while True:
      ret, frame = cap.read()         if
not ret:
         print("Error: Failed to capture frame.")            break
cv2.imshow('Original Video', frame)         slow_frame =
cv2.resize(frame, None,
fx=slow_factor,fy=slow_factor,interpolation=cv2.INTER_LINEAR)
cv2.imshow('Slow Motion', slow_frame)
      fast_frame = cv2.resize(frame, None, fx=fast_factor, fy=fast_factor,
interpolation=cv2.INTER_LINEAR)         cv2.imshow('Fast Motion',
fast_frame)         if cv2.waitKey(1) & 0xFF == ord('q'):
break      cap.release()
cv2.destroyAllWindows() video_path = 0 display_video_slow_fast(video_path)
```

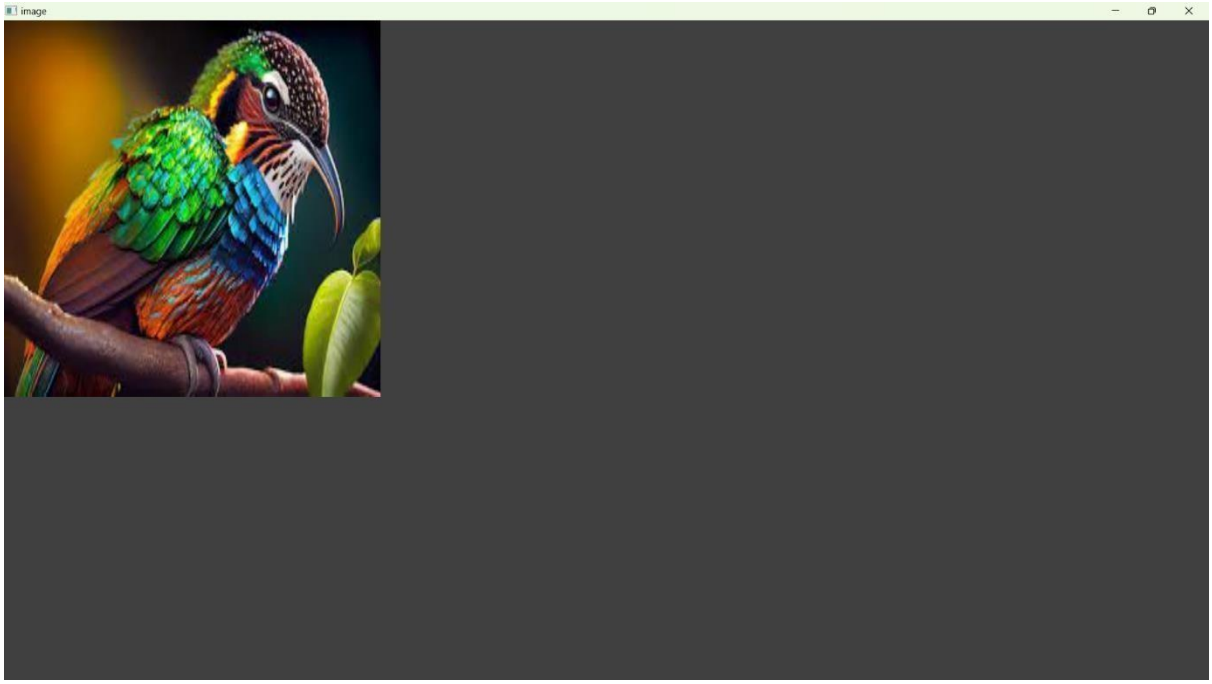
OUTPUT :

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/vempa/Downloads/BIRD.jpg",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\vempa\Downloads\BIRD2.jpg" src = cv2.imread(path)
window_name = 'Image'

image = cv2.rotate(src, cv2.ROTATE_90_COUNTERCLOCKWISE) cv2.imshow(window_name, image) cv2.waitKey(0) **INPUT:**

OUTPUT:

10.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\vempa\Downloads\BIRD2.jpg" src = cv2.imread(path)
window_name = 'Image' image =
cv2.rotate(src, cv2.ROTATE_180)
cv2.imshow(window_name, image) cv2.waitKey(0)
```

INPUT:

OUTPUT:

11. Perform Affine Transformation on the image.

AIM : To Perform Affine Transformation on the image.

PROGRAM :

```
import cv2 import numpy as
np

img = cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg")
rows,cols,_ = img.shape pts1 = np.float32([[50,50],[200,50],[50,200]])
pts2 = np.float32([[10,100],[200,50],[100,250]]) M =
cv2.getAffineTransform(pts1,pts2) dst =
cv2.warpAffine(img,M,(cols,rows)) cv2.imshow("Affine Transform", dst)
cv2.waitKey(0) cv2.destroyAllWindows()
```

INPUT:

OUTPUT:

12. Perform Perspective Transformation on the image. AIM : To Perform

Perspective Transformation on the image **PROGRAM :**

```
import cv2 import numpy as
np

img = cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg") rows,cols,ch = img.shape

pts1 = np.float32([[56,65],[368,52],[28,387],[389,390]]) pts2 =
np.float32([[100,50],[300,0],[0,300],[300,300]]) M =
cv2.getPerspectiveTransform(pts1,pts2) dst =
cv2.warpPerspective(img,M,(cols, rows)) cv2.imshow('Transformed
Image', dst) cv2.waitKey(0) cv2.destroyAllWindows()
```

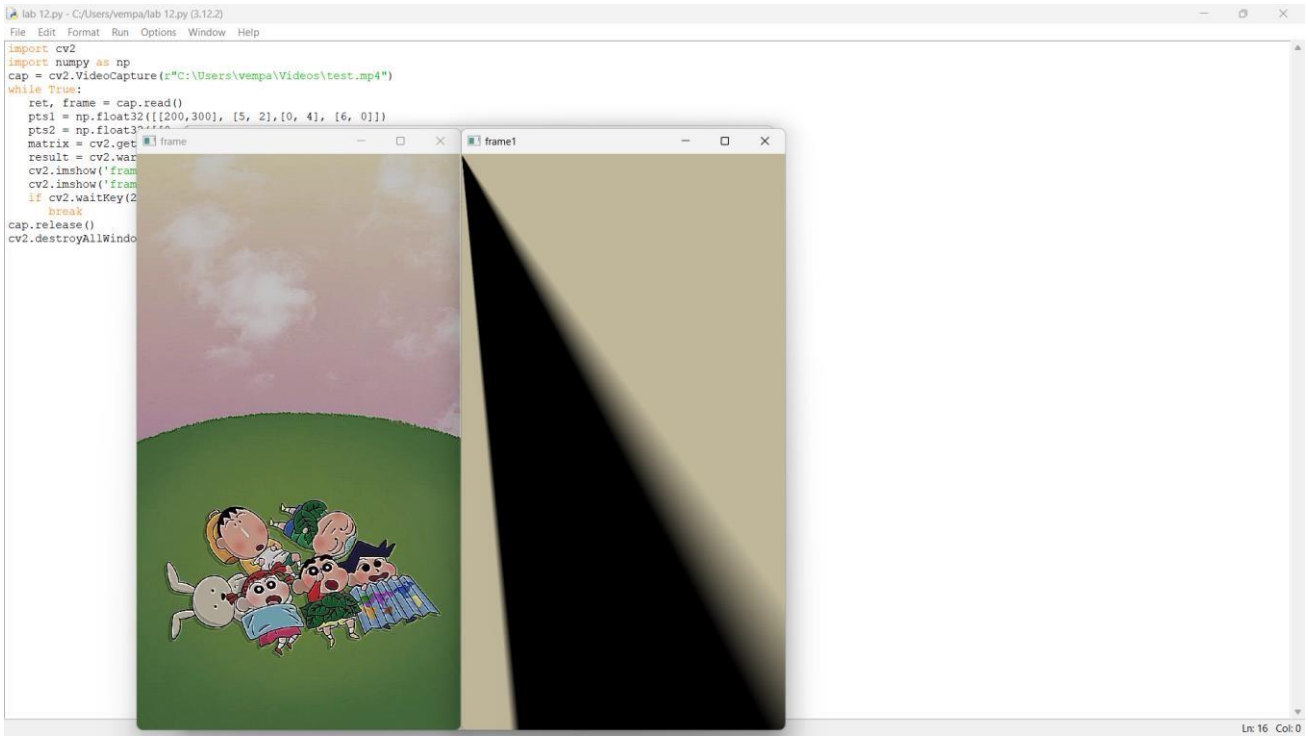
INPUT:

OUTPUT :

13. Perform Perspective Transformation on the Video

PROGRAM:

```
import cv2 import numpy as np cap =
cv2.VideoCapture(r"C:\Users\vempa\Videos\test.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

PROGRAM:

```
import cv2 import numpy as np im_src =
cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg")
```

pts_src = np.array([[141, 131], [480, 159], [493, 630],[64, 601]]) im_dst

= cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg")

pts_dst = np.array([[318, 256],[534, 372],[316, 670],[73, 473]]) h, status =

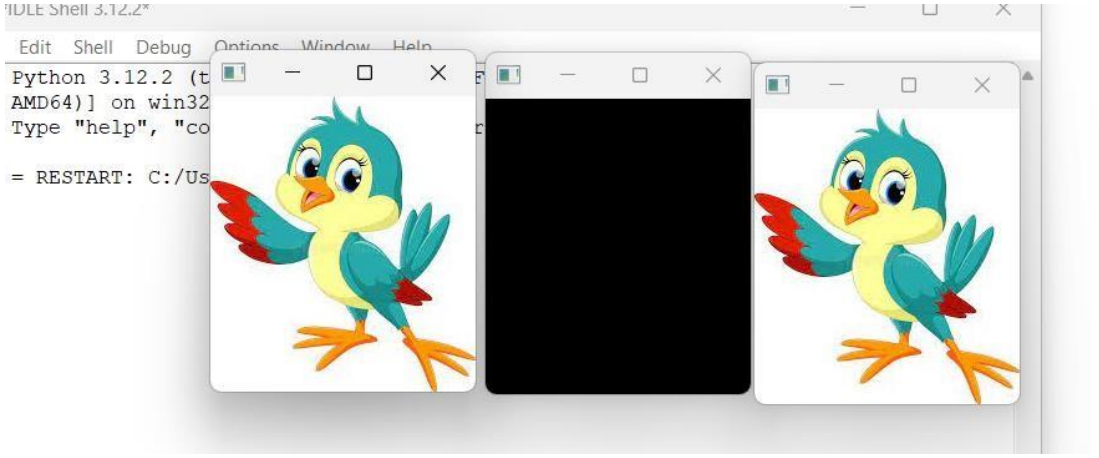
cv2.findHomography(pts_src, pts_dst) im_out =

cv2.warpPerspective(im_src, h, (im_dst.shape[1],im_dst.shape[0]))

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 img1 = cv2.imread(r"C:/Users/vempa/Downloads/HD-

wallpaper-tom-and-jerry-cartoons.jpg") img2 =

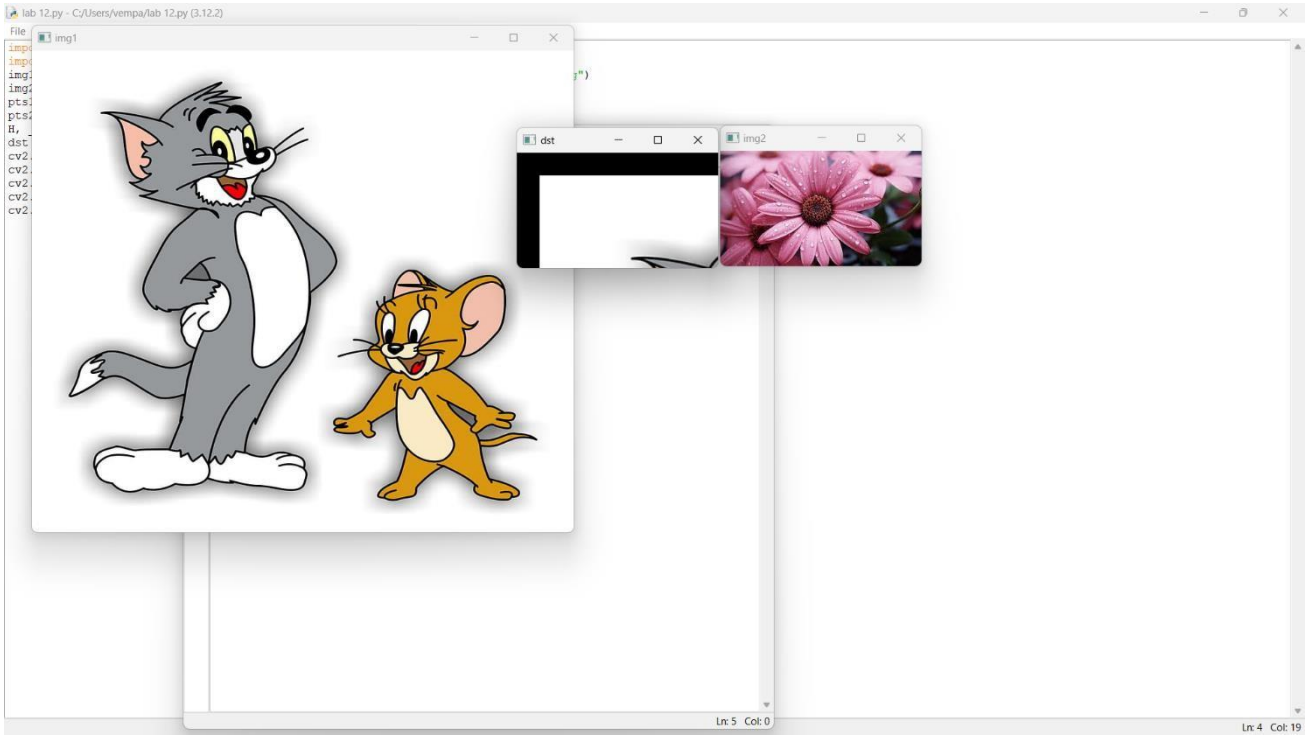
cv2.imread(r"C:\Users\vempa\Downloads\LAB4.jpg") pts1 = np.array([[50, 50], [200, 50], [50,

200], [200, 200]]) pts2 = np.array([[100, 100], [300, 100], [100, 300], [300, 300]]) H, _ =

cv2.findHomography(pts1, pts2) dst = cv2.warpPerspective(img1, H, (img2.shape[1],

img2.shape[0])) 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

Read the input image image_path =

r"C:\Users\vempa\Downloads\BIRD2.jpg" original_image =

cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

Check if the image is successfully loaded if

original_image is None:

print("Error: Could not load the image.") else:

Apply Gaussian blur to reduce noise and improve edge detection

blurred_image = cv2.GaussianBlur(original_image, (5, 5), 0)

Apply Canny edge detection edges = cv2.Canny(blurred_image, 50, 150)

Adjust the threshold values as needed

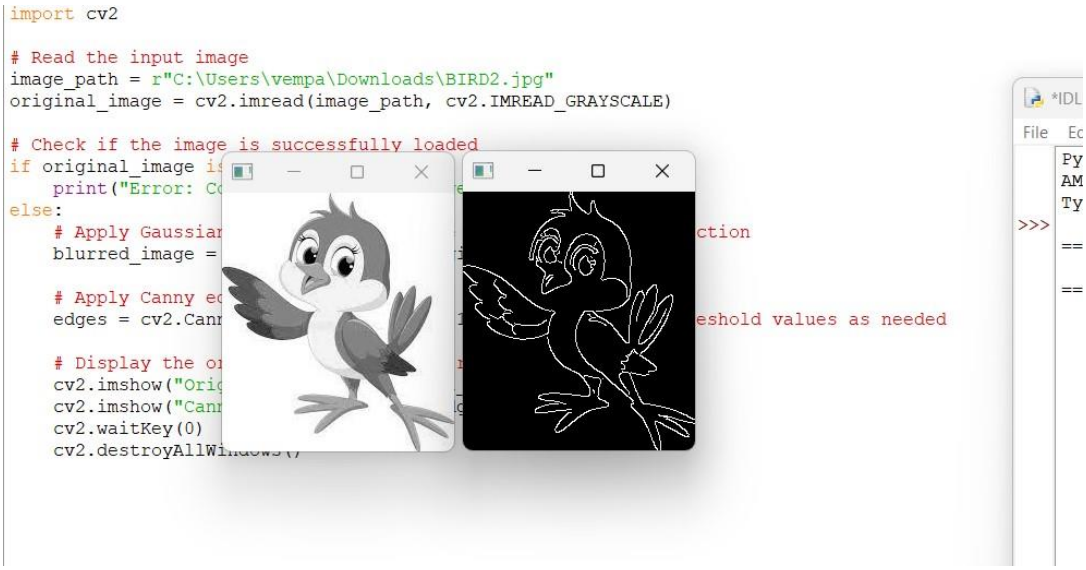
Display the original image and the result

cv2.imshow("Original Image", original_image)

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

cv2.waitKey(0) cv2.destroyAllWindows()

OUTPUT:



17. Perform Edge detection using Sobel Matrix along X axis

PROGRAM:

import cv2 img =

cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg")

cv2.imshow('Original', img) cv2.waitKey(0) img_gray =

cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

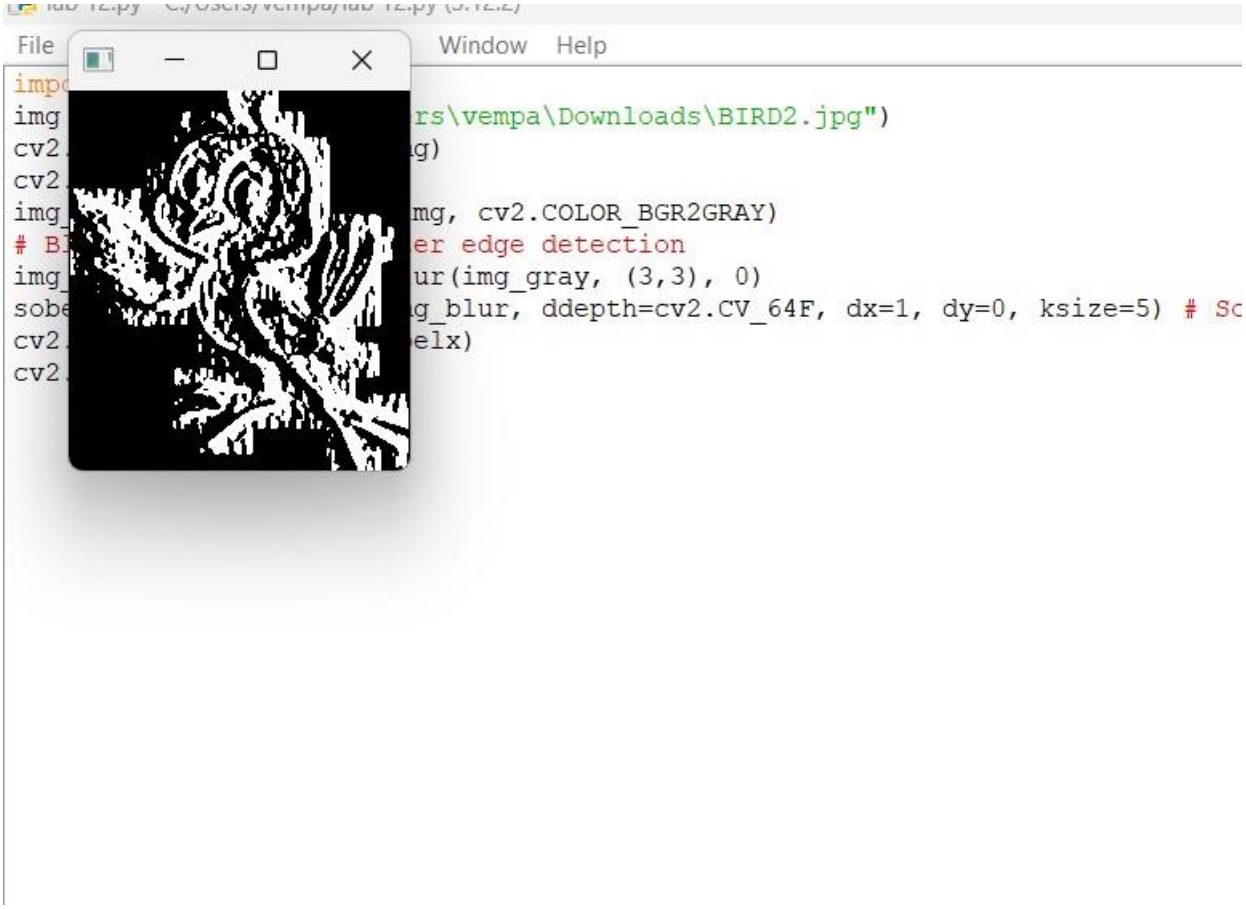
Blur the image for better edge detection img_blur

= 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(r"C:\Users\vempa\Downloads\BIRD2.jpg")

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(r"C:\Users\vempa\Downloads\BIRD2.jpg")

Display original image cv2.imshow('Original', img)

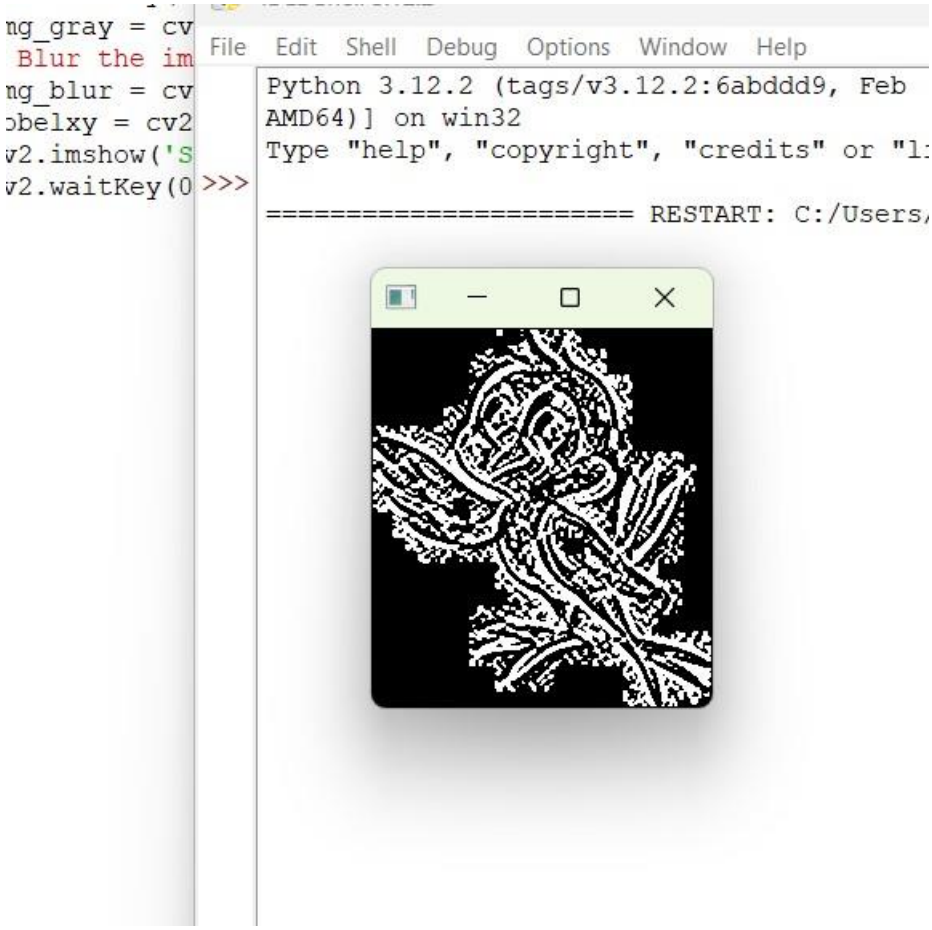
cv2.waitKey(0) img_gray = cv2.cvtColor(img,

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(r"C:\Users\vempa\Downloads\HD-wallpaper-tom-and-jerry-cartoons.jpg") gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

kernel = np.array([[0,1,0], [1,-8,1], [0,1,0]])

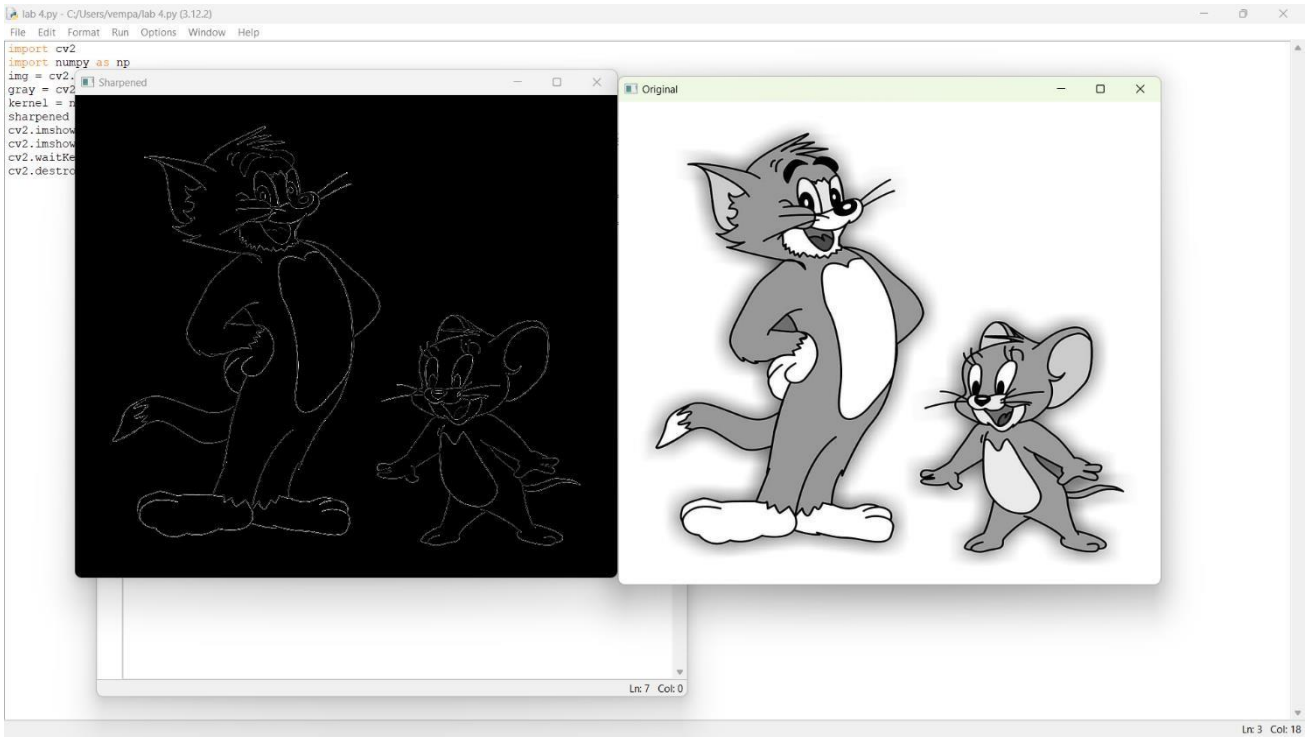
sharpened = cv2.filter2D(gray, -1, kernel)

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(r"C:\Users\vempa\Downloads\BIRD2.jpg") gray

= cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

kernel = np.array([[0,1,0], [1,-4,1], [0,1,0]])

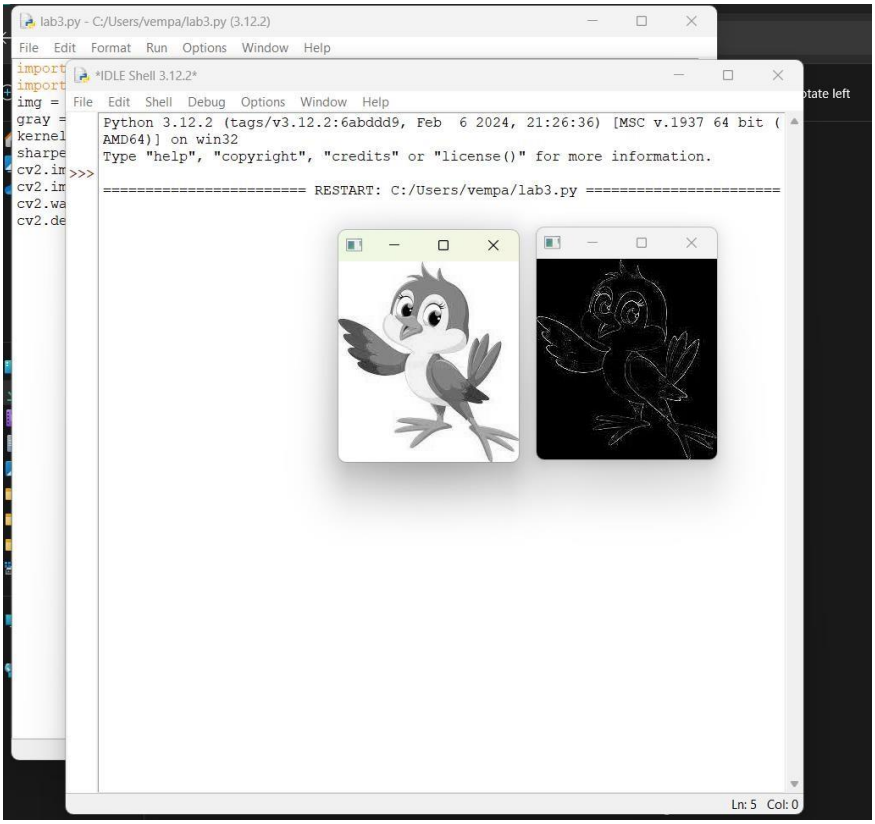
sharpened = cv2.filter2D(gray, -1, kernel)

cv2.imshow('Original', gray)

cv2.imshow('Sharpened', sharpened)

cv2.waitKey(0) cv2.destroyAllWindows()

OUTPUT:



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

PROGRAM:

import cv2 import numpy as np img =

cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.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() **OUTPUT:**

23. Perform Sharpening of Image using unsharp masking.

PROGRAM:

import cv2 import numpy as np img =

cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.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,

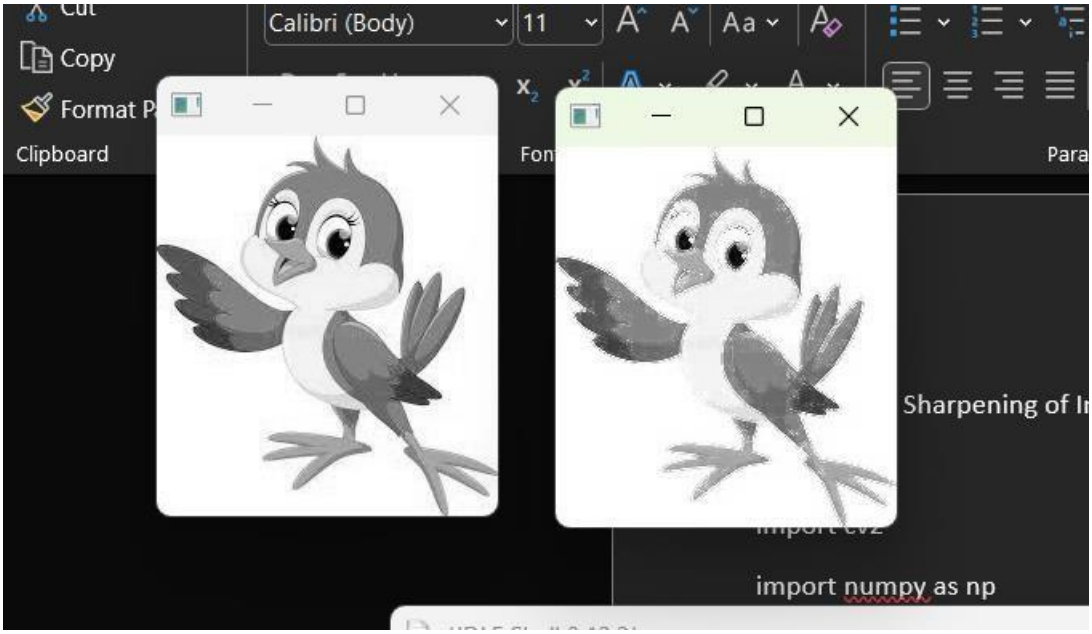
-1, laplacian_kernel) sharpened = cv2.add(gray,

laplacian) cv2.imshow('Original Image', gray)

cv2.imshow('Sharpened Image', sharpened)

cv2.waitKey(0) cv2.destroyAllWindows()

OUTPUT:

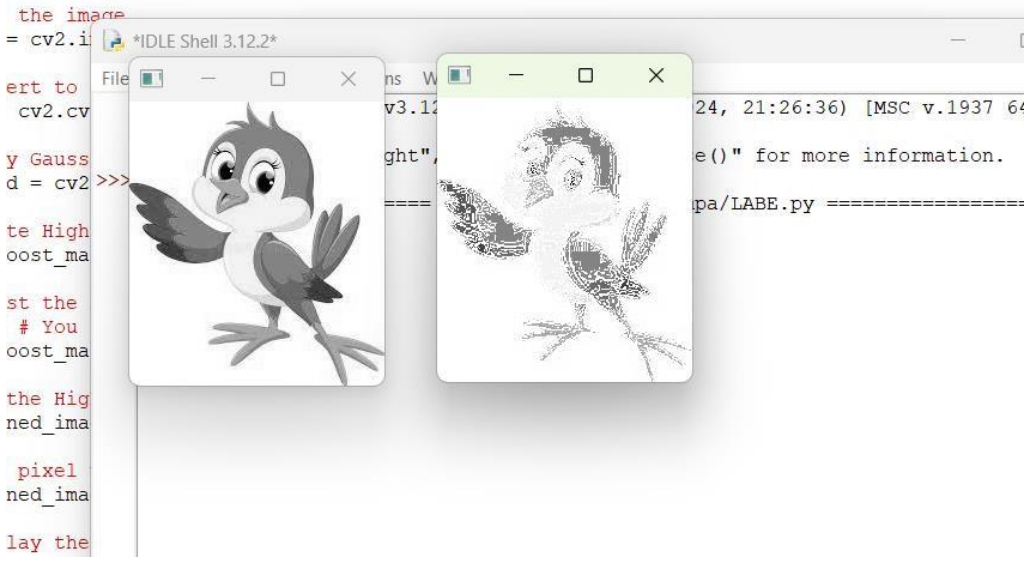


24. Perform Sharpening of Image using High-Boost Masks.

PROGRAM:

```
import cv2 import numpy as np # Load the image image =
cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg")
# Convert to grayscale gray = cv2.cvtColor(image,
cv2.COLOR_BGR2GRAY)
# Apply Gaussian blur blurred =
cv2.GaussianBlur(gray, (5, 5), 0)
high_boost_mask = gray - blurred
A = 2  # You can experiment with different values of A
high_boost_mask = A * high_boost_mask # Add the
High-Boost Mask to the Original Image
sharpened_image = cv2.add(gray, high_boost_mask)
sharpened_image = np.clip(sharpened_image, 0, 255)
cv2.imshow('Original Image', gray)
cv2.imshow('Sharpened Image', sharpened_image)
cv2.waitKey(0) cv2.destroyAllWindows()
```

OUTPUT :



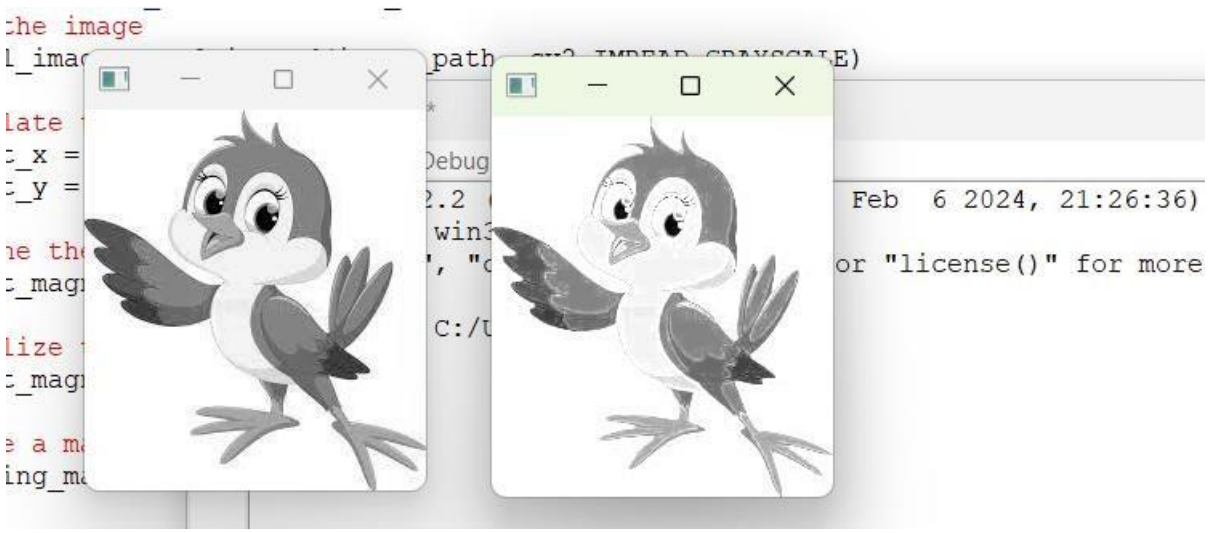
25. Perform Sharpening of Image using Gradient masking

PROGRAM:

import cv2 import numpy as np def image_sharpening_gradient(image_path, alpha=1.5):

Load the image original_image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

Calculate the gradient using Sobel operators gradient_x = cv2.Sobel(original_image, cv2.CV_64F, 1, 0, ksize=3) gradient_y = cv2.Sobel(original_image, cv2.CV_64F, 0, 1, ksize=3) gradient_magnitude = np.sqrt(gradient_x**2 + gradient_y**2) gradient_magnitude = cv2.normalize(gradient_magnitude, None, 0, 255, cv2.NORM_MINMAX) sharpening_mask = original_image + alpha * gradient_magnitude sharpening_mask = np.clip(sharpening_mask, 0, 255) sharpening_mask = np.uint8(sharpening_mask) cv2.imshow('Original Image', original_image) cv2.imshow('Sharpening Mask', sharpening_mask) cv2.waitKey(0) cv2.destroyAllWindows() image_sharpening_gradient(r"C:\Users\vempa\Downloads\BIRD2.jpg", alpha=1.5) **OUTPUT** :



26. Insert water marking to the image using OpenCV.

PROGRAM:

import cv2 import numpy as np def add_watermark(input_image_path, output_image_path, watermark_path, position=(0, 0), alpha=0.7):

Load the original image original_image = cv2.imread(input_image_path) # Load the watermark image with an alpha channel watermark = cv2.imread(watermark_path, cv2.IMREAD_UNCHANGED)

Extract the alpha channel from the watermark alpha_channel = watermark[:, :, 3] / 255.0 # Resize the watermark to fit the desired position h, w = original_image.shape[:2] watermark_resized = cv2.resize(watermark, (w // 5, h // 5)) # Define the region of interest (ROI) for the watermark placement roi = original_image[-watermark_resized.shape[0]:, -watermark_resized.shape[1]:] blended = cv2.addWeighted(roi, 1 - alpha, watermark_resized[:, :, :3], alpha, 0)

Update the original image with the blended ROI original_image[-watermark_resized.shape[0]:, -watermark_resized.shape[1]:] = blended cv2.imwrite(output_image_path, original_image) cv2.imshow('Watermarked Image', original_image) cv2.waitKey(0) cv2.destroyAllWindows()

add_watermark(r"C:\Users\vempa\Downloads\BIRD2.jpg", r"C:\Users\vempa\Downloads\download (1).png", "C:\drive\OneDrive\Pictures\Screenshots\Screenshot (275).png", position=(0, 0), alpha=0.7)

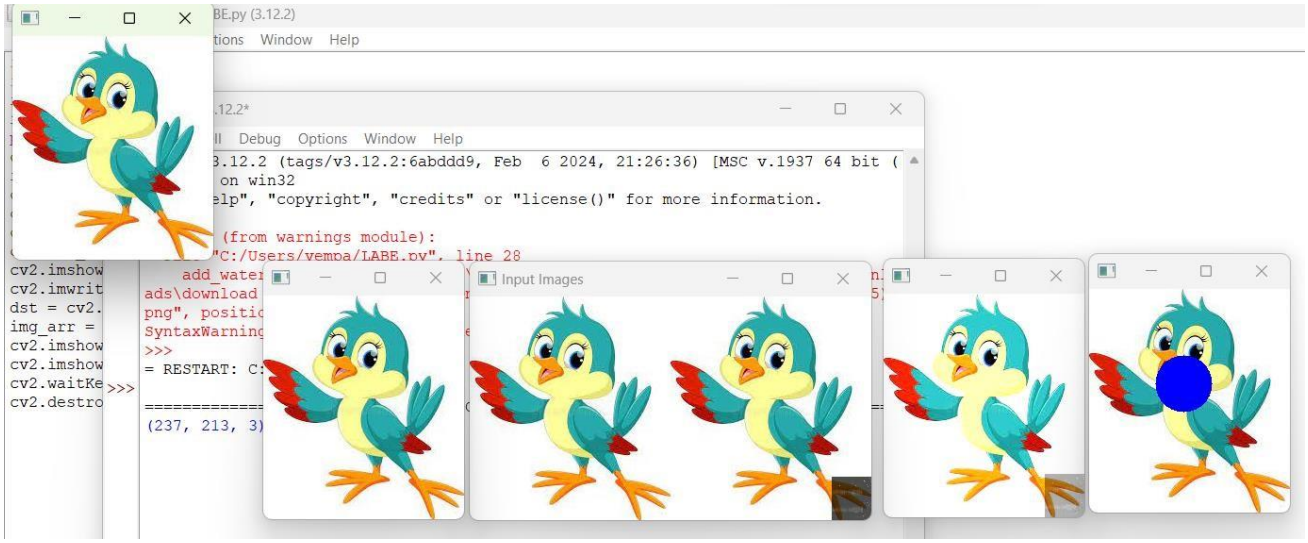
OUTPUT:

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

PROGRAM:

```
import cv2 import numpy as np image =
cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg") img2 =
cv2.imread(r"C:\Users\vempa\Downloads\download (1).png")
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:

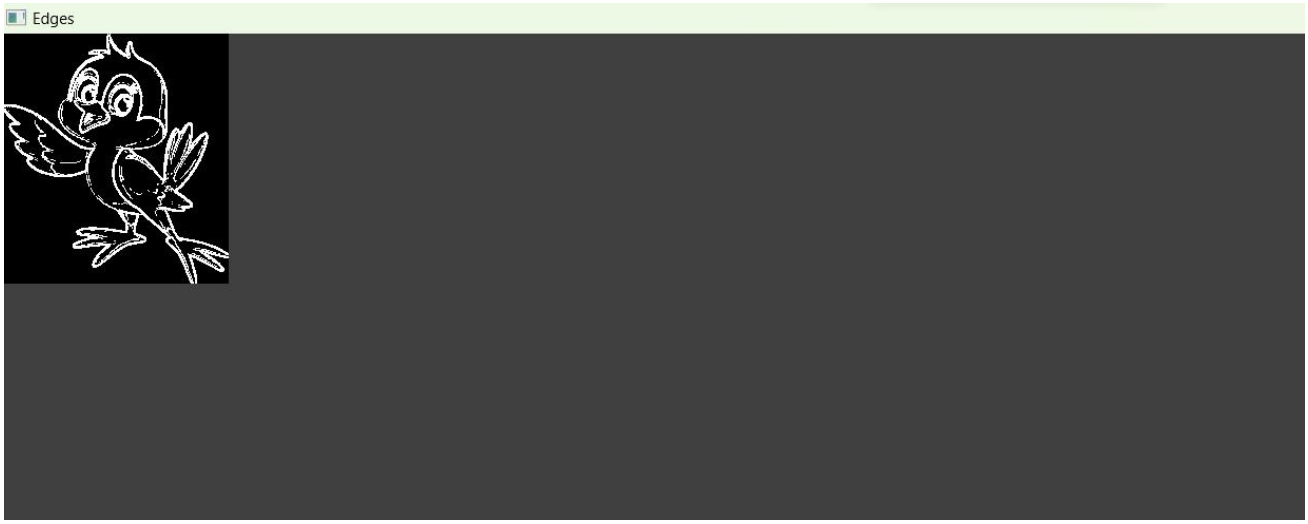


28. Find the boundary of the image using Convolution kernel for the given image

PROGRAM:

```
import cv2 import numpy as np img =
cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.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()
```

OUTPUT:

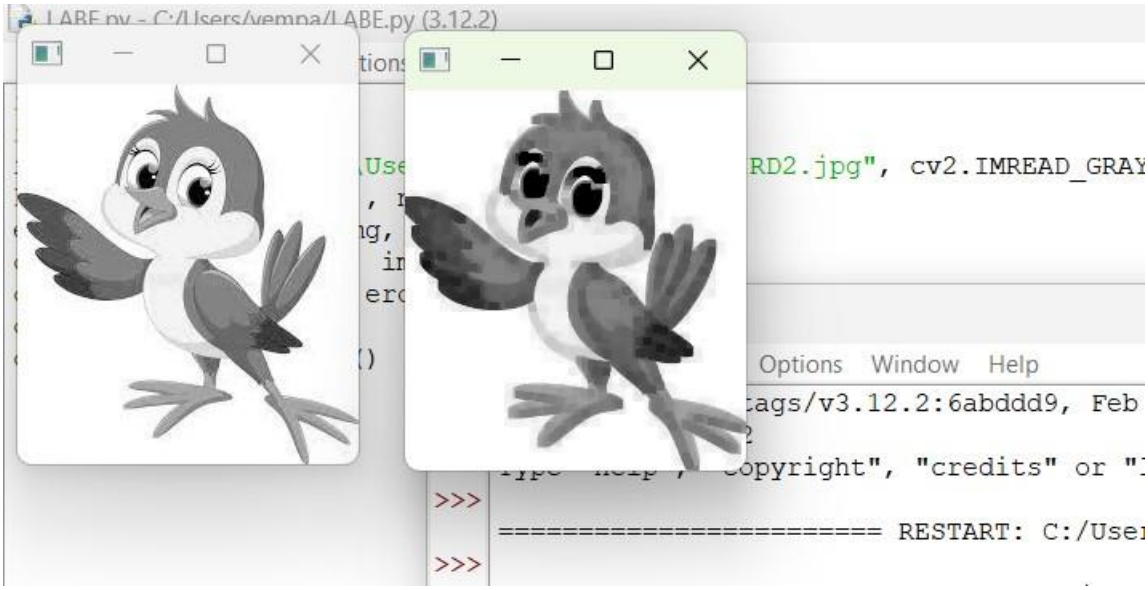


29. Morphological operations based on OpenCV using Erosion technique

PROGRAM:

```
import cv2 import numpy as np img =
cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.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()
```

OUTPUT:



30. Morphological operations based on OpenCV using Dilation technique

PROGRAM:

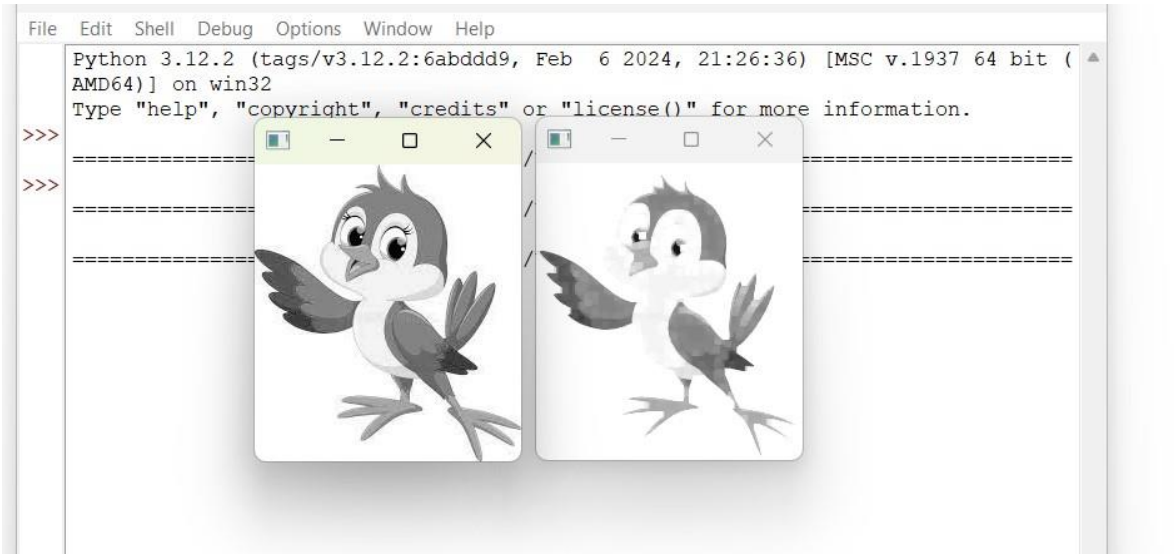
import cv2 import numpy as np img =

cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.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() **OUTPUT:**



31. Morphological operations based on OpenCV using Opening technique.

PROGRAM:

import cv2 import numpy as np img = cv2.imread

(r"C:\Users\vempa\Downloads\BIRD2.jpg", cv2.IMREAD_GRAYSCALE)

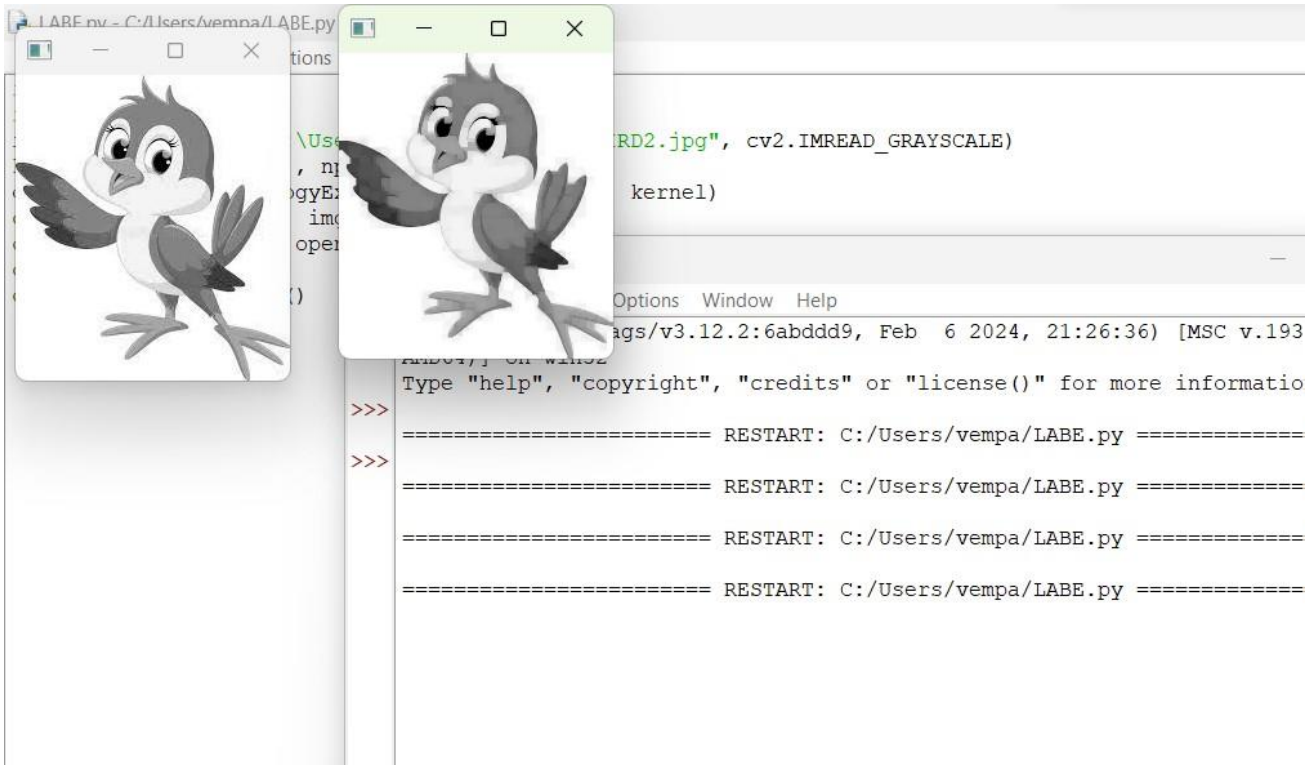
kernel = np.ones((5,5), np.uint8) opening =

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(r"C:\Users\vempa\Downloads\BIRD2.jpg", cv2.IMREAD_GRAYSCALE)

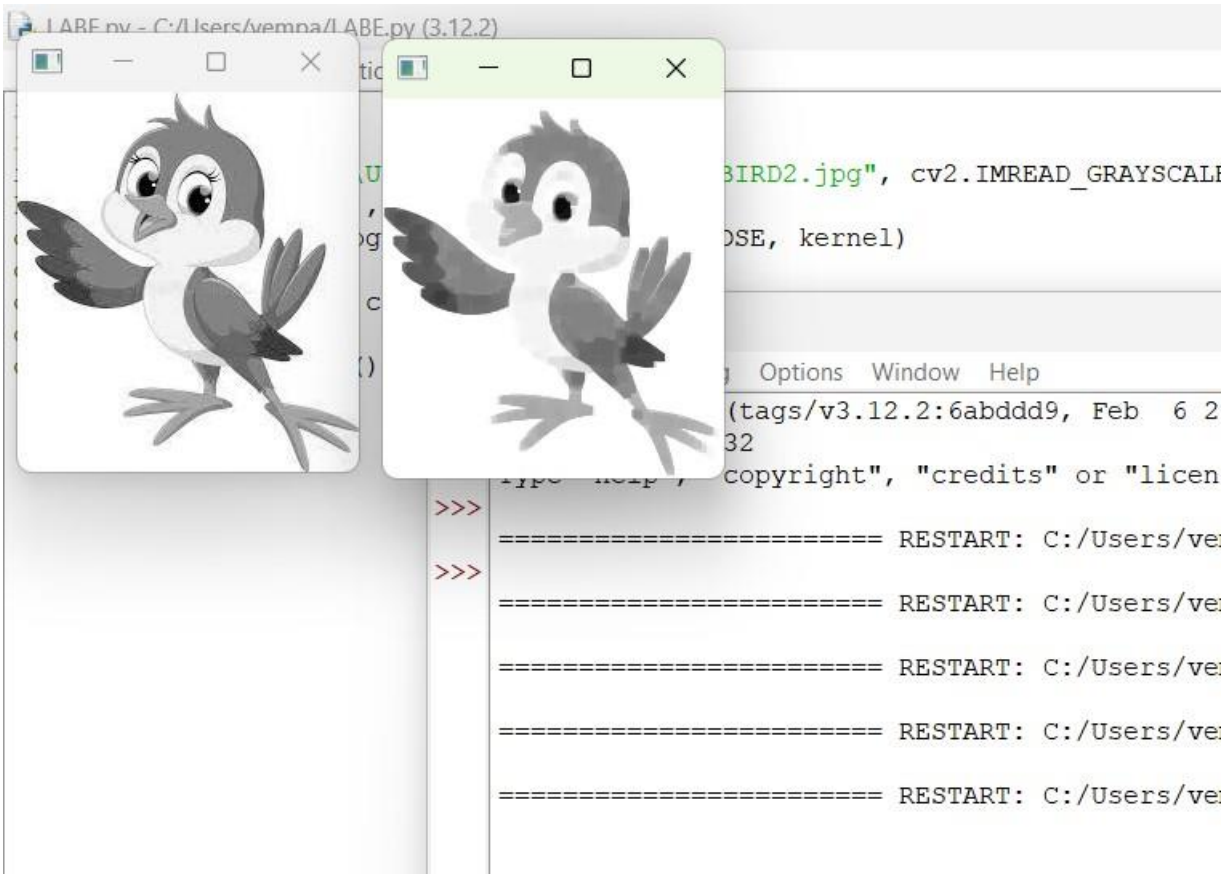
kernel = np.ones((5,5), np.uint8) closing =

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

cv2.imshow("Original", img) cv2.imshow("Closing", closing)

cv2.waitKey(0)

cv2.destroyAllWindows() **OUTPUT**:



33. Morphological operations based on OpenCV using Morphological Gradient technique

PROGRAM:

import cv2 import numpy as np img =

cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.jpg", cv2.IMREAD_GRAYSCALE)

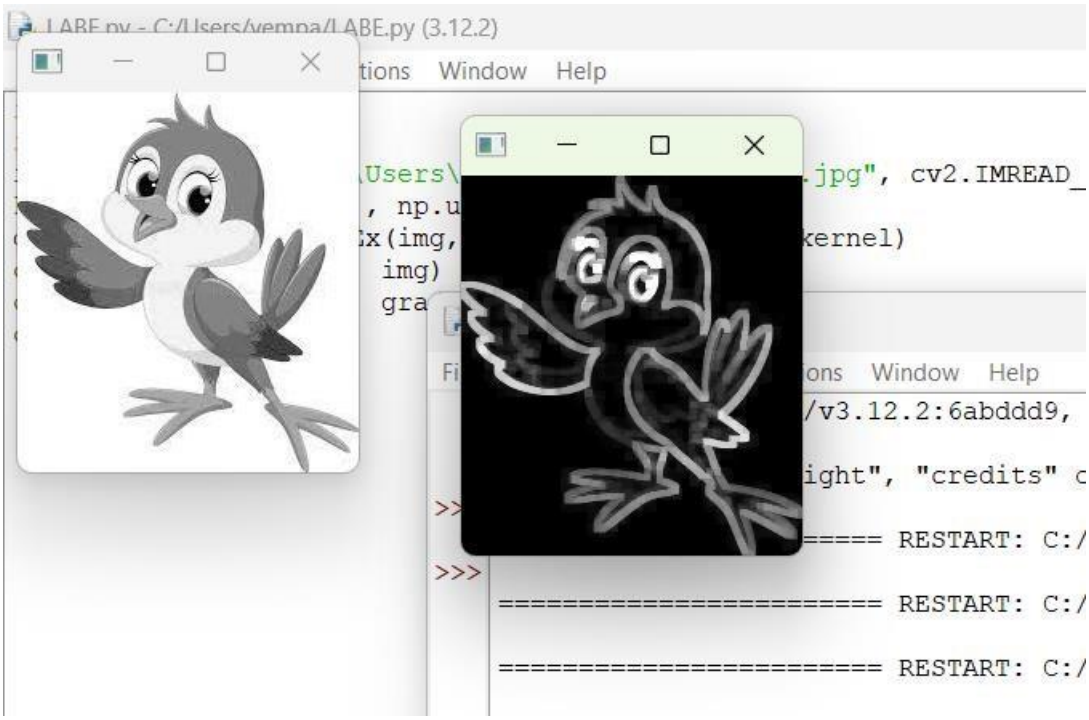
kernel = np.ones((5,5), np.uint8)

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(r"C:\Users\vempa\Downloads\BIRD2.jpg", cv2.IMREAD_GRAYSCALE)

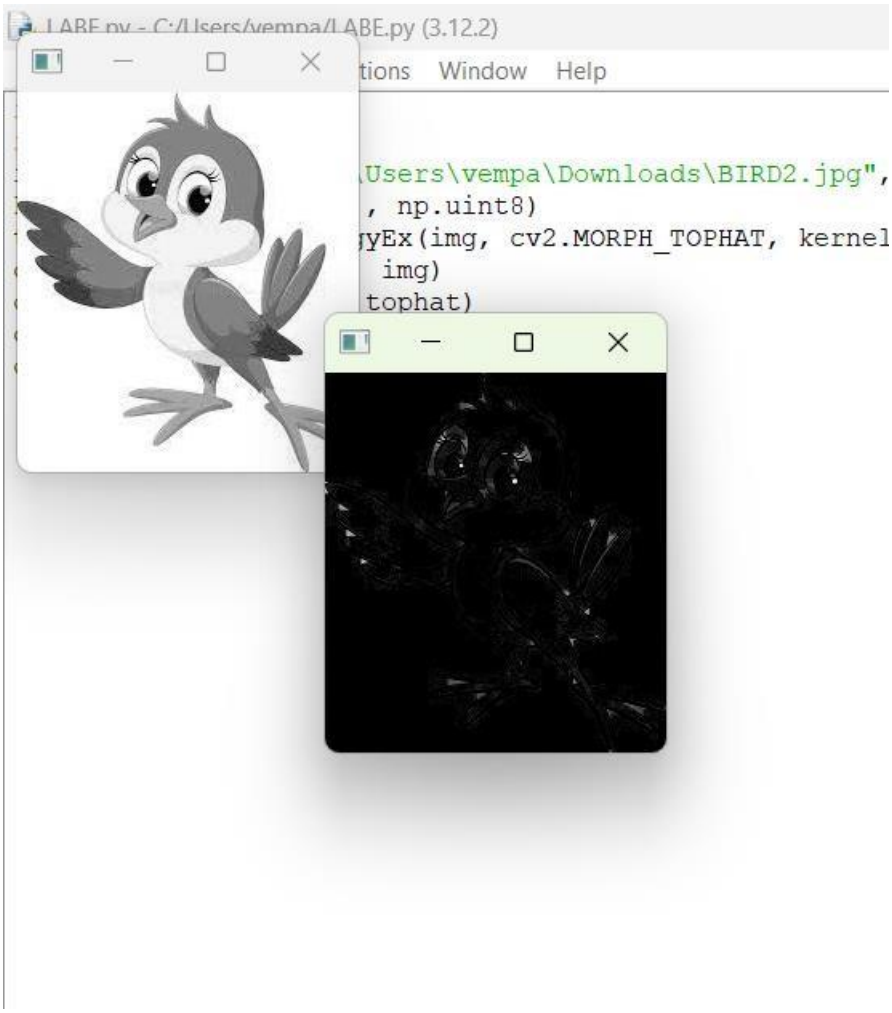
kernel = np.ones((5,5), np.uint8) tophat =

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(r"C:\Users\vempa\Downloads\BIRD2.jpg", cv2.IMREAD_GRAYSCALE)

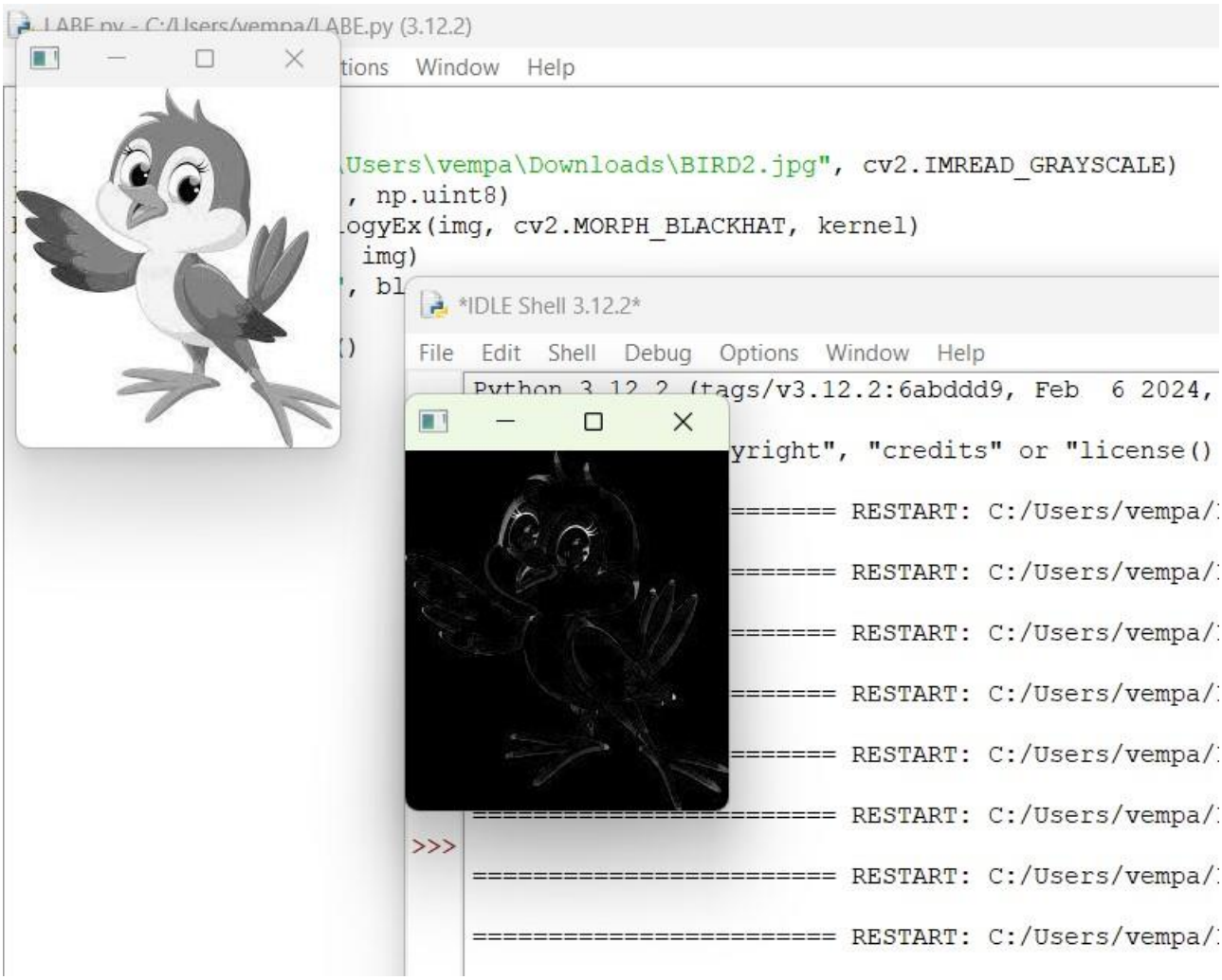
kernel = np.ones((5,5), np.uint8) blackhat =

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:\drive\OneDrive\Documents\watch-cascade.xml") img = cv2.imread("C:\drive\OneDrive\Pictures\Screenshots\Screenshot 2024-02-26 092427.png") 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:**

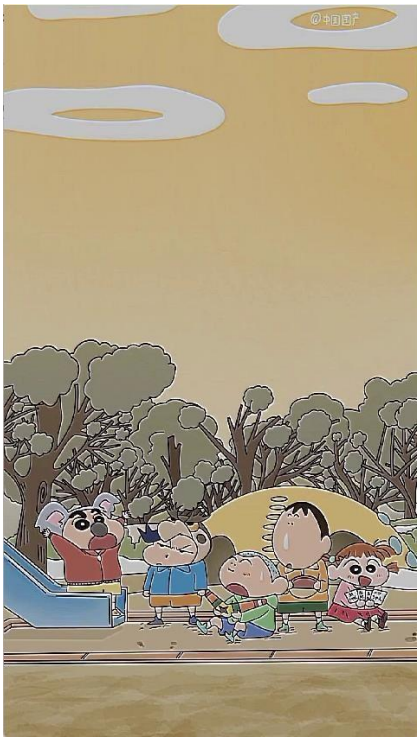
37. Using Opencv play Video in Reverse mode.

PROGRAM:

```
import cv2 cap =
cv2.VideoCapture(r"C:\Users\vempa\Videos\test.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(30) & 0xFF == ord('q'):
```

```
    break
current_frame -= 1
cap.release()
cv2.destroyAllWindows()
```

OUTPUT:

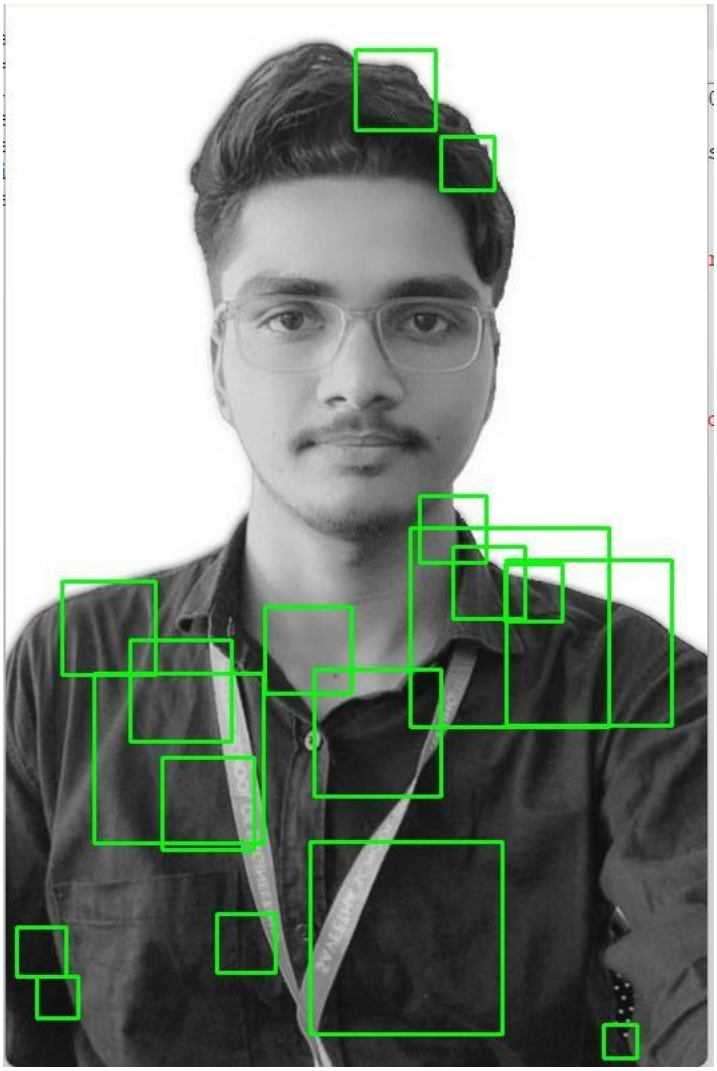


38. Face Detection using Opencv

PROGRAM:

```
import cv2 img = cv2.imread("C:\drive\OneDrive\Pictures\Screenshots\Screenshot 2024-02-21
123000.png") gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) face_cascade =
cv2.CascadeClassifier("C:\drive\OneDrive\Documents\watch-cascade.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:



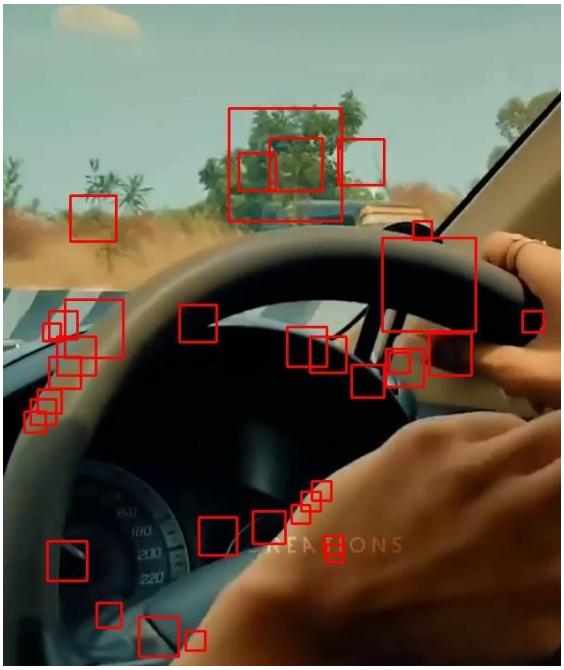
39. Vehicle Detection in a Video frame using OpenCV

PROGRAM:

```
import cv2 car_cascade = cv2.CascadeClassifier("C:\\drive\\OneDrive\\Documents\\watch-

cascade.xml")

cap = cv2.VideoCapture("C:\\drive\\OneDrive\\Pictures\\Slide Shows\\Ram's\\WA-VID-20200720-
9aa8edb7.mp4") while

True:

   ret, frame = cap.read()

if not ret:        break

   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:

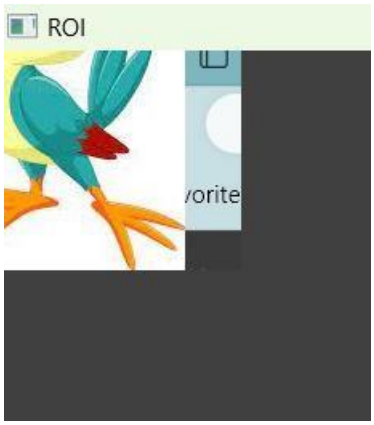


40. Draw Rectangular shape and extract objects

PROGRAM:

```
import cv2 img =
cv2.imread(r"C:\Users\vempa\Downloads\BIRD2.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:


ROI
vorite