

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

PROGRAM:

```
import cv2

import numpy as np

img = cv2.imread(r"C:\Users\91824\OneDrive\anime.jpeg")

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()
```

INPUT:



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/91824/OneDrive/affilitate.jpeg")

img = cv2.resize(img,(255, 255))

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

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()
```

INPUT:



OUTPUT:



23. Perform Sharpening of Image using unsharp masking.

PROGRAM:

```
import cv2

import numpy as np

img = cv2.imread(r"C:\Users\91824\OneDrive\todo.jpeg")

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()
```

INPUT:



OUTPUT:



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

PROGRAM:

```
import cv2

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 = r"C:\Users\91824\OneDrive\naruto.jpeg"
cv2.imwrite(filename, resized_img)

cv2.imshow("Resized Input Image", resized_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

INPUT:



OUTPUT:

25. Perform Sharpening of Image using Gradient masking

PROGRAM:

```
import cv2

import numpy as np

a = cv2.imread(r"C:\Users\91824\OneDrive\boruto.jpg", cv2.IMREAD_GRAYSCALE)

if a is None:

    print("Error: Unable to load image.")

    exit()

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

a1 = cv2.filter2D(a, -1, Lap)

a2 = np.uint8(a1)

cv2.imshow("Laplacian Convolution", a2)

cv2.waitKey(0)

cv2.destroyAllWindows()

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

a3 = cv2.filter2D(a, -1, lap)

a4 = np.uint8(a3)

cv2.imshow("Laplacian Edge Enhancement", a4)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:



26. Insert water marking to the image using OpenCV.

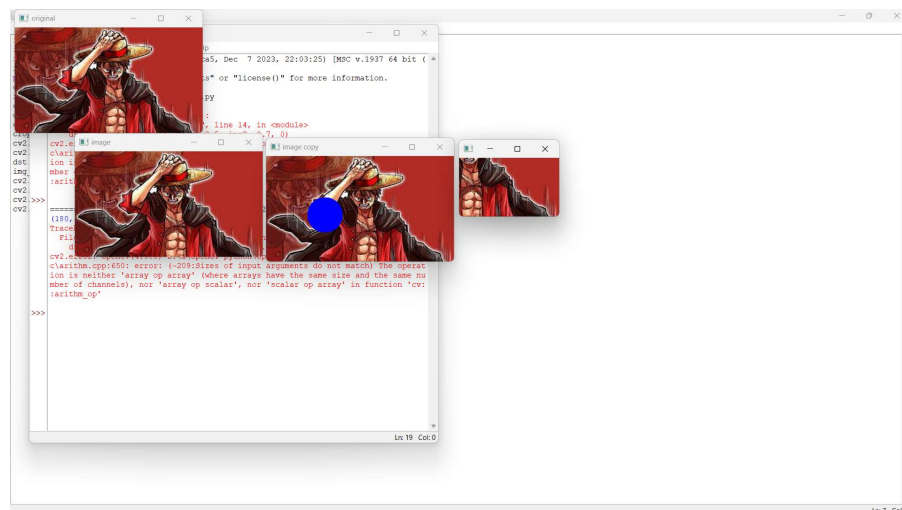
PROGRAM:

```
import cv2

img = cv2.imread(r"C:\Users\91824\OneDrive\yuzi.jpeg.jpg")
wm = cv2.imread(r"C:\Users\91824\OneDrive\eswar2.jpeg.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_top = max(0, top_y)
roi_bottom = min(h_img, bottom_y)
roi_left = max(0, left_x)
roi_right = min(w_img, right_x)
roi = img[roi_top:roi_bottom, roi_left:roi_right]
wm = cv2.resize(wm, (roi.shape[1], roi.shape[0]))
result = cv2.addWeighted(roi, 1, wm, 0.3, 0)
img[roi_top:roi_bottom, roi_left:roi_right] = result
cv2.imshow("Watermarked Image", img)
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\91824\OneDrive\eswar2.jpeg.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()
```

INPUT:



OUTPUT:



29. Morphological operations based on OpenCV using Erosion technique

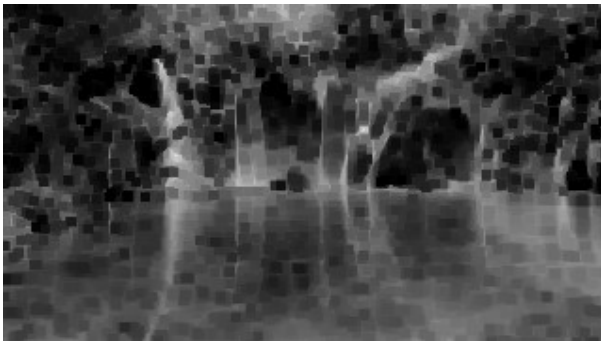
PROGRAM:

```
import cv2
import numpy as np
img = cv2.imread(r"C:\Users\91824\OneDrive\water.jpeg", 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()
```

INPUT:



OUTPUT:



30. Morphological operations based on OpenCV using Dilation technique

PROGRAM:

```
import cv2  
  
import numpy as np  
  
img = cv2.imread(r"C:\Users\91824\OneDrive\flower.jpeg", 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()
```

INPUT:



OUTPUT :

