Date : 06/08/2025

FCV WEEK3 SUBMISSION

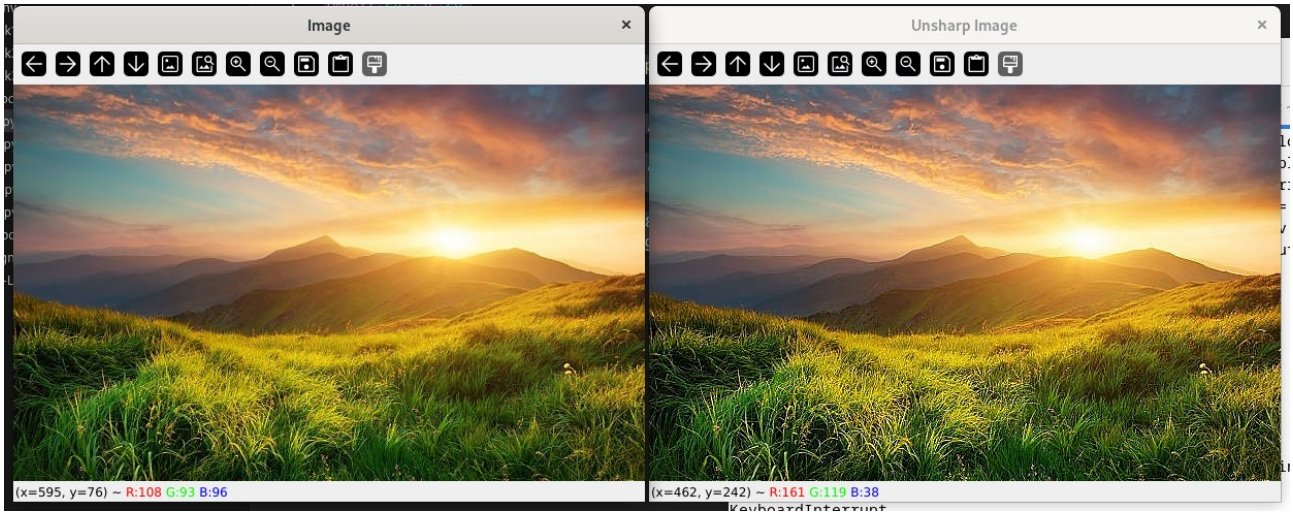
Name: Davasam Karthikeya Section: AIMLB Reg No: 230962326

1. Write a program to read an image and perform unsharp masking.

Code:-

```
import cv2 as cv
import numpy as np

img = cv.imread('Week1/img.jpg').astype(np.float32)


img_blurred = cv.GaussianBlur(img, (3,3), 0)

unsharp_img = cv.addWeighted(img, 2.3, img_blurred, -1.3, 0)
unsharp_img = np.clip(unsharp_img, 0, 255).astype(np.uint8)

cv.imshow('Image', img.astype(np.uint8))
cv.imshow('Unsharp Image', unsharp_img)

cv.waitKey(0)
cv.destroyAllWindows()
```

Output:-



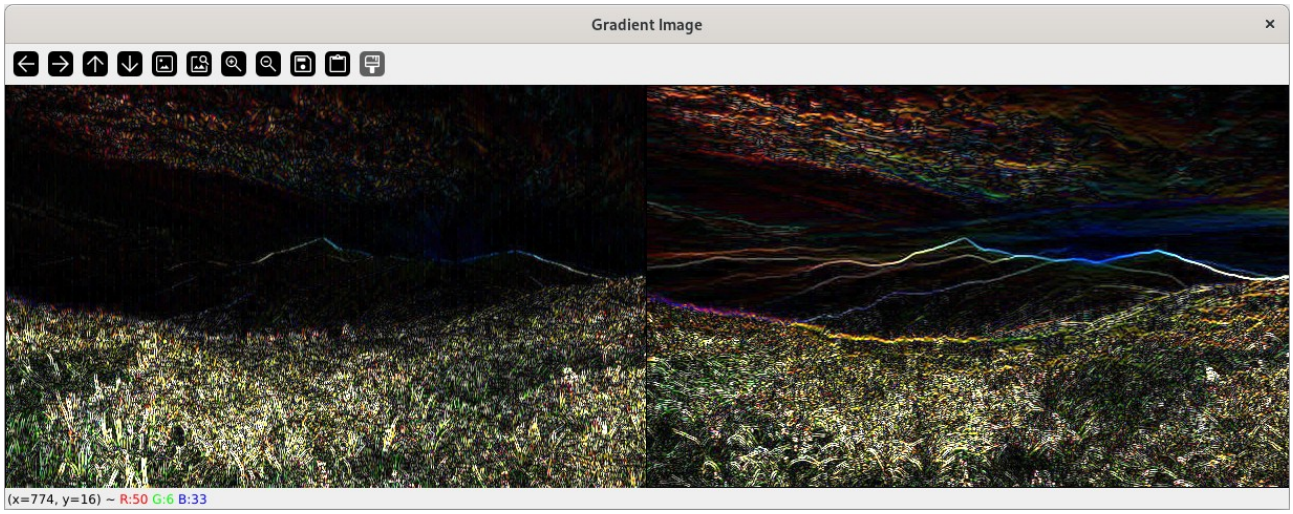
2. Write a program to obtain gradient of an image.

Code:-

```
import cv2 as cv
import numpy as np

img = cv.imread('Week1/img.jpg')

grad_x = cv.convertScaleAbs(cv.Sobel(img, cv.CV_64F, 1, 0, ksize=3))
grad_y = cv.convertScaleAbs(cv.Sobel(img, cv.CV_64F, 0, 1, ksize=3))
```

```
cv.imshow('Gradient Image', np.hstack([grad_x, grad_y]))

cv.waitKey(0)
cv.destroyAllWindows()
```

Output:-

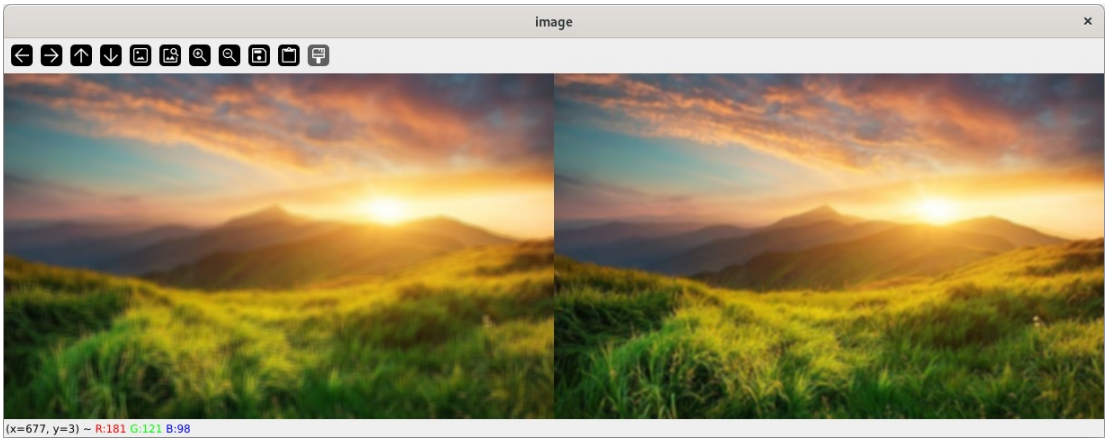


3. Write a program to compare box filter and gaussian filter image outputs.

Code:-

```
import cv2 as cv
import numpy as np

img = cv.imread('Week1/img.jpg')

box_blur = cv.blur(img, (7,7))
gauss_blur = cv.GaussianBlur(img, (7,7), 0)

cv.imshow('image', np.hstack([box_blur, gauss_blur]))

cv.waitKey(0)
cv.destroyAllWindows()
```

Output:-



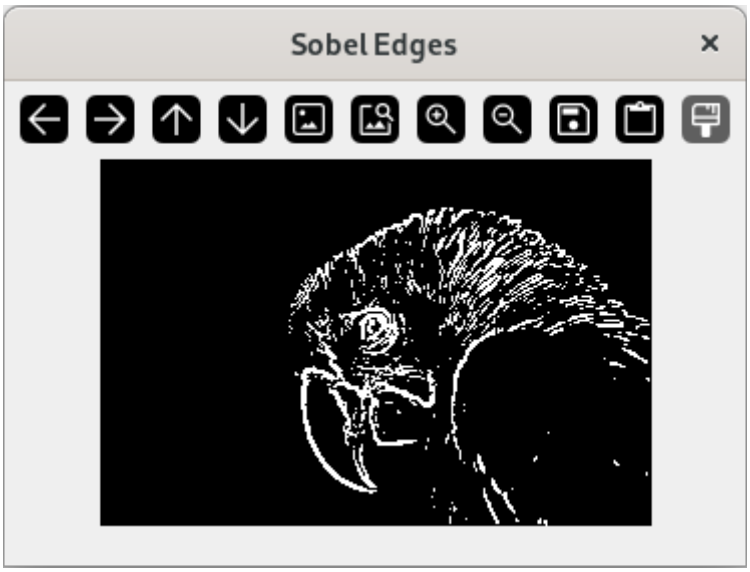
Date : 06/08/2025

4. Write a program to detect edges in a image.

Code:-

```
import cv2 as cv
import numpy as np


img = cv.imread('Week1/images.jpeg')

grad_x = cv.convertScaleAbs(cv.Sobel(img, cv.CV_64F, 1, 0, ksize=3))
grad_y = cv.convertScaleAbs(cv.Sobel(img, cv.CV_64F, 0, 1, ksize=3))

sobel_edges = cv.cvtColor(cv.addWeighted(grad_x, 0.5, grad_y, 0.5, 0),
cv.COLOR_BGR2GRAY)

_, sobel_edges = cv.threshold(sobel_edges, 100, 255, type=cv.THRESH_BINARY)

cv.imshow("Sobel Edges", sobel_edges)
cv.waitKey(0)
cv.destroyAllWindows()
```

Output:-



5. Implement Canny edge detection algorithm.

Code:-

```
import cv2
import numpy as np

img = cv2.imread('Week1/img.jpg')
height, width, _ = img.shape

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

gx = cv2.Sobel(np.float32(img), cv2.CV_64F, 1, 0, 3)
gy = cv2.Sobel(np.float32(img), cv2.CV_64F, 0, 1, 3)
mag, ang = cv2.cartToPolar(gx, gy, angleInDegrees=True)
```

Date : 06/08/2025

```
mag_max = np.max(mag)
weak_th = mag_max * 0.1
strong_th = mag_max * 0.5

for i_x in range(width):
    for i_y in range(height):
        grad_ang = (ang[i_y, i_x] + 22.5) % 180
        grad_ang = abs(
            grad_ang - 180) if abs(grad_ang) > 180 else abs(grad_ang)
        if grad_ang <= 45:
            neighb_1_x, neighb_1_y = i_x - 1, i_y
            neighb_2_x, neighb_2_y = i_x + 1, i_y
        elif grad_ang > 45 and grad_ang <= 90:
            neighb_1_x, neighb_1_y = i_x - 1, i_y - 1
            neighb_2_x, neighb_2_y = i_x + 1, i_y + 1
        elif grad_ang > 90 and grad_ang <= 135:
            neighb_1_x, neighb_1_y = i_x, i_y - 1
            neighb_2_x, neighb_2_y = i_x, i_y + 1
        elif grad_ang > 135 and grad_ang <= 180:
            neighb_1_x, neighb_1_y = i_x - 1, i_y + 1
            neighb_2_x, neighb_2_y = i_x + 1, i_y - 1

        if 0 <= neighb_1_x < width and 0 <= neighb_1_y < height:
            if mag[i_y, i_x] < mag[neighb_1_y, neighb_1_x]:
                mag[i_y, i_x] = 0
                continue
        if 0 <= neighb_2_x < width and 0 <= neighb_2_y < height:
            if mag[i_y, i_x] < mag[neighb_2_y, neighb_2_x]:
                mag[i_y, i_x] = 0

ids = np.zeros_like(img)
for i_x in range(width):
    for i_y in range(height):
        grad_mag = mag[i_y, i_x]
        if grad_mag < weak_th:
            mag[i_y, i_x] = 0
        elif strong_th > grad_mag >= weak_th:
            ids[i_y, i_x] = 1
        else:
            ids[i_y, i_x] = 2

cv2.imshow('Canny Edges', mag)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output:-