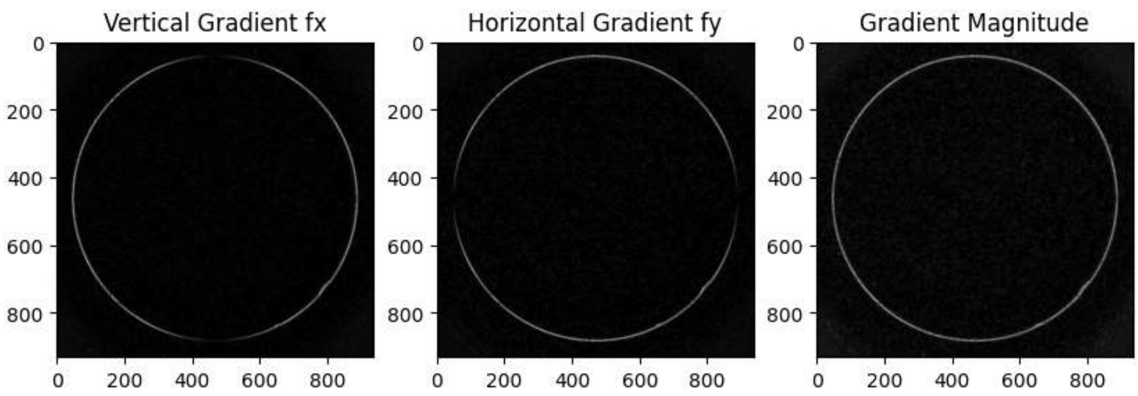
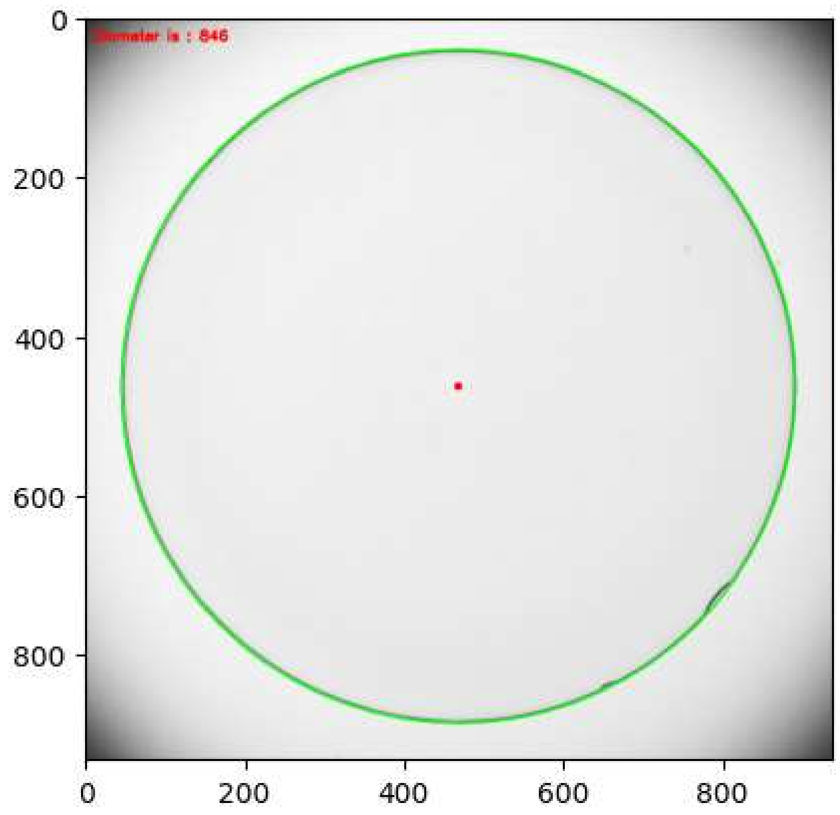
In [ ]:

```
#1
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('lens.JPG')
assert im is not None
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
sobel_x = cv2.Sobel(gray, cv2.CV_64F, 1, 0)
sobel_y = cv2.Sobel(gray, cv2.CV_64F, 0, 1)
magnitude = cv2.addWeighted(np.absolute(sobel_x), 1, np.absolute(sobel_y), 1, 0)

fig, ax = plt.subplots(1, 3, figsize=(10, 5))
ax[0].imshow(np.absolute(sobel_x).astype('uint8'), cmap='gray')
ax[0].set_title("Vertical Gradient fx")
ax[1].imshow(np.absolute(sobel_y).astype('uint8'), cmap='gray')
ax[1].set_title("Horizontal Gradient fy")
ax[2].imshow(magnitude.astype('uint8'), cmap='gray')
ax[2].set_title("Gradient Magnitude")
plt.show()

blurred = cv2.GaussianBlur(gray, (5, 5), 0)

edges = cv2.Canny(blurred, 50, 150)
y_cord, x_cord = np.where(edges != 0)
Center_X = int(np.mean(x_cord))
Center_Y = int(np.mean(y_cord))
distances = np.sqrt((x_cord - Center_X)**2 + (y_cord - Center_Y)**2)
diameter = int(2 * np.max(distances))
cv2.circle(img, (Center_X, Center_Y), 5, (0, 0, 255), -1)
cv2.circle(img, (Center_X, Center_Y), diameter // 2, (0, 255, 0), 2)
cv2.putText(img, f"Diameter is : {diameter}", (10, 30), cv2.FONT_ITALIC, 0.6, (0
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```

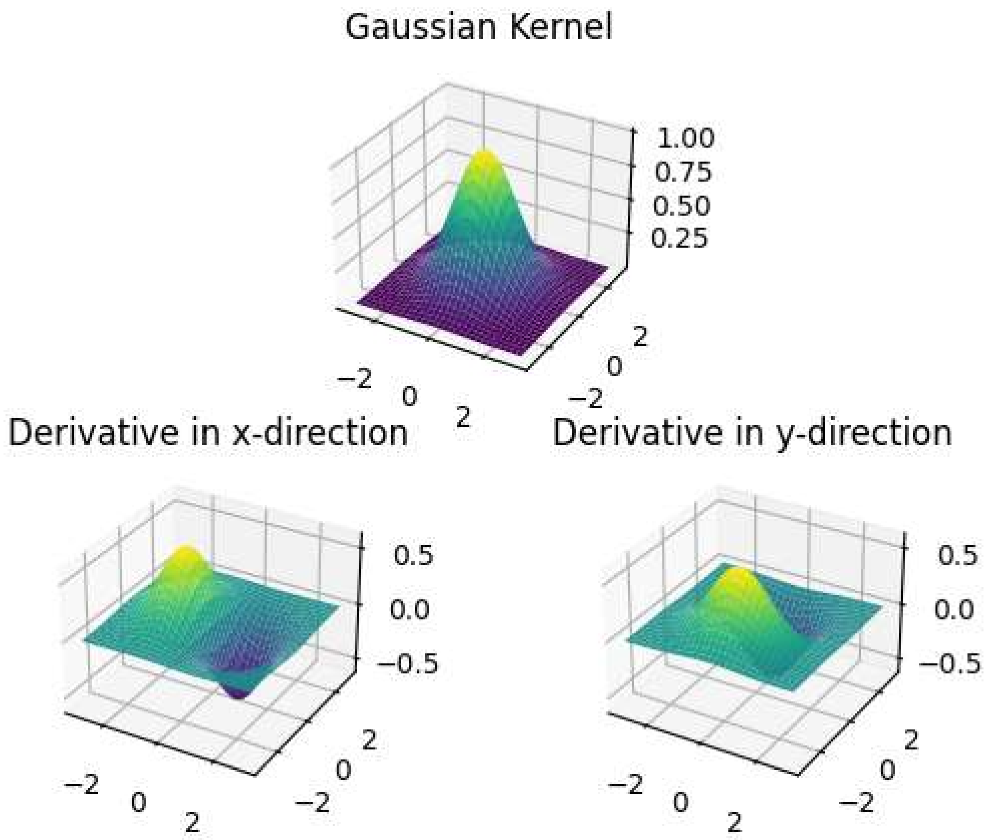





In [ ]:

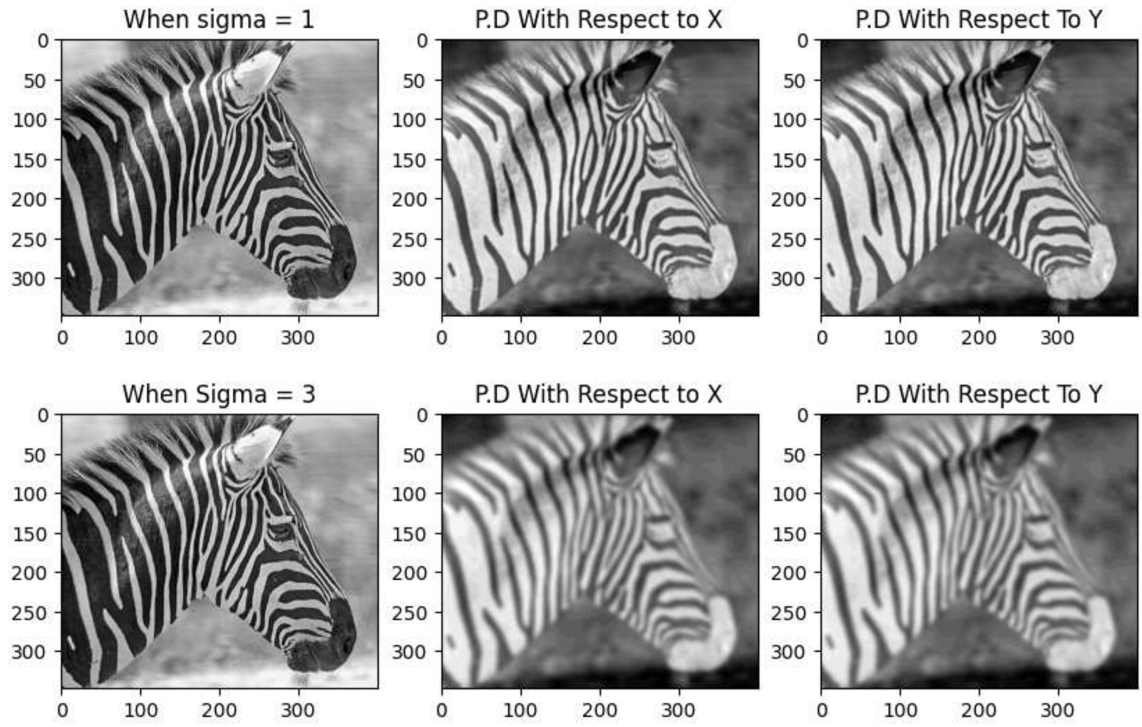
```
#2
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def gaussian(x, y, sigma):
    return np.exp(-(x**2 + y**2) / (2 * sigma**2))

def gaussian_dx(x, y, sigma):
    return -x / sigma**2 * gaussian(x, y, sigma)

def gaussian_dy(x, y, sigma):
    return -y / sigma**2 * gaussian(x, y, sigma)

def gaussian_dxy(x, y, sigma):
    return x * y / sigma**4 * gaussian(x, y, sigma)

sigma = 1
x = np.arange(-3, 3.1, 0.1)
y = np.arange(-3, 3.1, 0.1)
X, Y = np.meshgrid(x, y)
Zx = gaussian_dx(X, Y, sigma)
Zy = gaussian_dy(X, Y, sigma)

fig = plt.figure()
ax = fig.add_subplot(2, 1, 1, projection='3d')
ax.plot_surface(X, Y, gaussian(X, Y, sigma), cmap='viridis')
ax.set_title('Gaussian Kernel')

ax = fig.add_subplot(2, 2, 3, projection='3d')
ax.plot_surface(X, Y, Zx, cmap='viridis')
ax.set_title('Derivative in x-direction')

ax = fig.add_subplot(2, 2, 4, projection='3d')
```

```
ax.plot_surface(X, Y, Zy, cmap='viridis')
ax.set_title('Derivative in y-direction')

plt.show()
```




In [ ]:
```
#3
import numpy as np
import cv2 as cv
from scipy import ndimage
import matplotlib.pyplot as plt

im = cv.imread('zebrahead.jpg', cv.IMREAD_GRAYSCALE)
assert im is not None

sigma = 1
kernel = np.zeros((5*sigma, 5*sigma))
for i in range(5*sigma):
    for j in range(5*sigma):
        x = i - 2*sigma
        y = j - 2*sigma
        kernel[i, j] = 1/(2*np.pi*sigma**2) * np.exp(-(x**2+y**2)/(2*sigma**2))
dKdx = ndimage.convolve(im, -kernel/sigma**2)
dKdy = ndimage.convolve(im, -kernel.T/sigma**2)
fig, ax = plt.subplots(1, 3, figsize=(10, 5))

ax[0].imshow(im, cmap='gray')
ax[0].set_title('When sigma = 1')
ax[1].imshow(dKdx, cmap='gray')
ax[1].set_title('P.D With Respect to X')
ax[2].imshow(dKdy, cmap='gray')
ax[2].set_title('P.D With Respect To Y')
plt.show()

sigma = 3
```

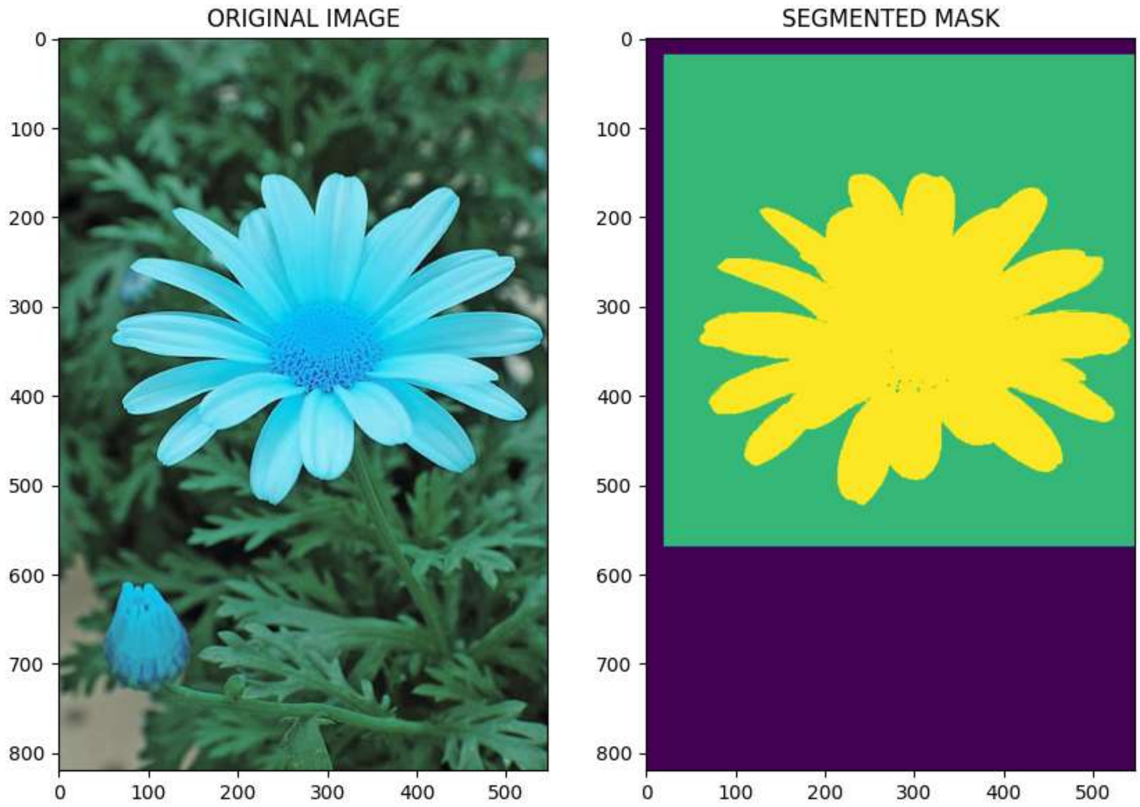
```
kernel = np.zeros((5*sigma, 5*sigma))
for i in range(5*sigma):
    for j in range(5*sigma):
        x = i - 2*sigma
        y = j - 2*sigma
        kernel[i, j] = 1/(2*np.pi*sigma**2) * np.exp(-(x**2+y**2)/(2*sigma**2))
dKdx = ndimage.convolve(im, -kernel/sigma**2)
dKdy = ndimage.convolve(im, -kernel.T/sigma**2)
fig, ax = plt.subplots(1, 3, figsize=(10, 5))

ax[0].imshow(im, cmap='gray')
ax[0].set_title('When Sigma = 3')
ax[1].imshow(dKdx, cmap='gray')
ax[1].set_title('P.D With Respect to X')
ax[2].imshow(dKdy, cmap='gray')
ax[2].set_title('P.D With Respect To Y')
plt.show()
```

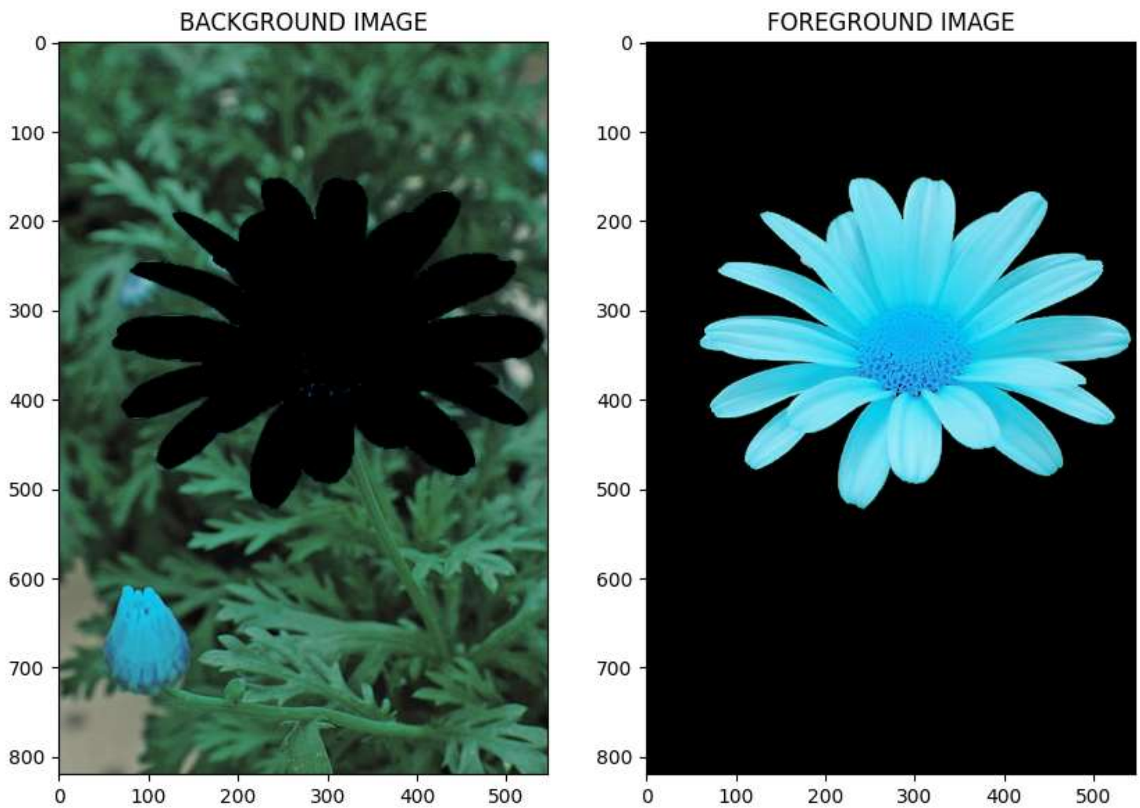



In [ ]:

```
#4
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np

img= cv.imread('daisy.jpg')
assert img is not None

rectangle = (20,20,550,550)
mask = np.zeros(img.shape[:2], np.uint8)
background = np.zeros((1,65), np.float64)
cv.grabCut(img, mask, rectangle, background, None, 5, cv.GC_INIT_WITH_RECT)
newmask = np.where((mask == cv.GC_FGD) | (mask == cv.GC_PR_FGD), 1, 0).astype('u
foreground = cv.bitwise_and(img, img, mask=newmask)
background = cv.bitwise_and(img, img, mask=1 - newmask)
blurred_background = cv.GaussianBlur(background,(31,31),0)
enhanced_image = cv.addWeighted(foreground, 1, blurred_background, 0.8, 0)
fig ,ax = plt.subplots (2,2, figsize= (10,20))
ax[0,0].imshow(img,cmap = 'gray')
```

```
ax[0,0].set_title('ORIGINAL IMAGE')
ax[0,1].imshow(mask)
ax[0,1].set_title('SEGMENTED MASK')
ax[1,1].imshow(foreground)
ax[1,1].set_title('FOREGROUND IMAGE')
ax[1,0].imshow(background)
ax[1,0].set_title('BACKGROUND IMAGE')
plt.figure(figsize=(12,6))
plt.subplot(1, 2, 1), plt.imshow(img[:,:,::-1]), plt.title('ORIGINAL IMAGE')
plt.axis('off')
plt.subplot(1, 2, 2), plt.imshow(enhanced_image[:,:,::-1]), plt.title('ENHANCED
plt.axis('off')
plt.show()
```


ORIGINAL IMAGE
SEGMENTED MASK


BACKGROUND IMAGE
FOREGROUND IMAGE

ORIGINAL IMAGE

ENHANCED IMAGE