Name : Munasinghe M.M.R.H.

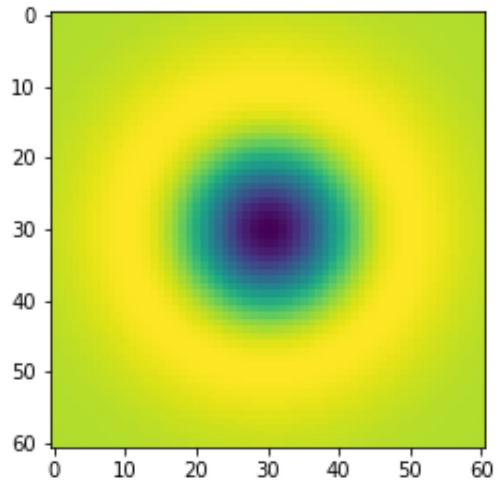
Index No. : 190399L

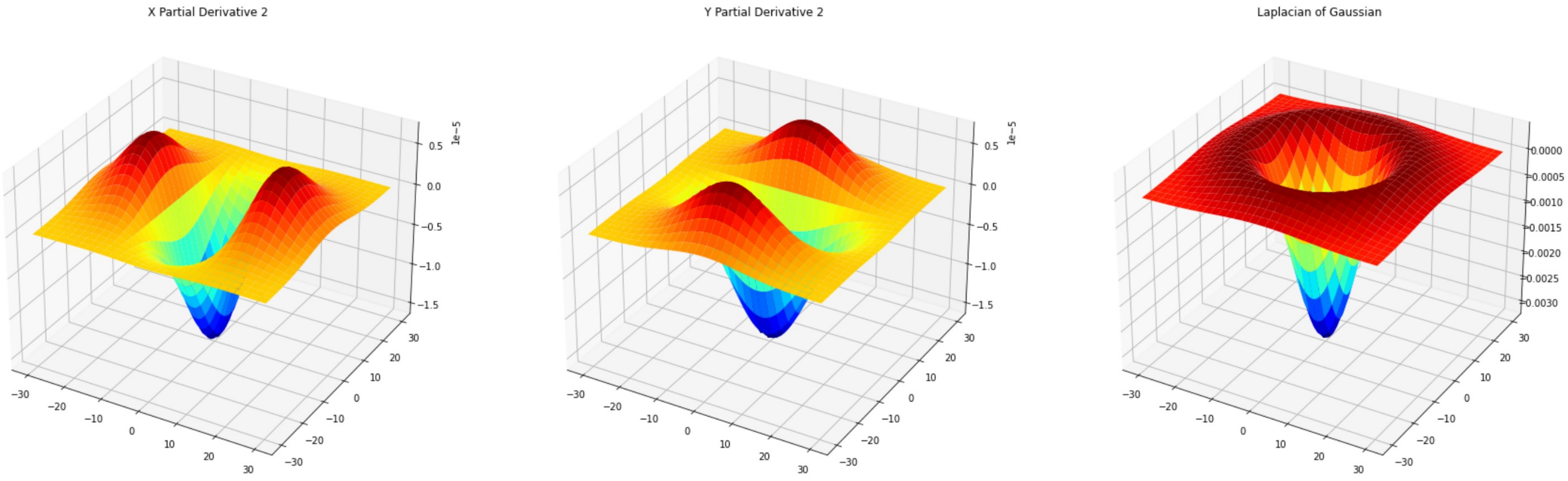
In [ ]:

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt
from matplotlib import cm
import math as m
from mpl_toolkits.mplot3d.axes3d import get_test_data
```

In [ ]:

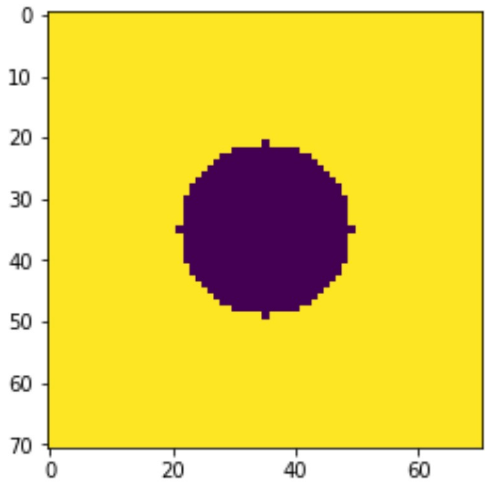
```
x=np.linspace(-10,10, num=100)
y=np.linspace(-10,10, num=100)

sigma = 10
half_width = 3*sigma
X_ = np.arange(-half_width, half_width + 1, 1)
Y_ = np.arange(-half_width, half_width + 1, 1)
X, Y = np.meshgrid(X_,Y_)
Partial_Der_X = np.exp(-(X**2 + Y**2)/(2*sigma**2))*(X**2/sigma**2-1)/(2*m.pi*sigma**4)
Partial_Der_Y = np.exp(-(X**2 + Y**2)/(2*sigma**2))*(Y**2/sigma**2-1)/(2*m.pi*sigma**4)
Laplacian = sigma**2*(Partial_Der_X + Partial_Der_Y)

plt.imshow(Laplacian)
fig = plt.figure(figsize=(30,30))
ax = fig.add_subplot(1, 3, 1, projection='3d')
surf = ax.plot_surface(X, Y, Partial_Der_X,  cmap=cm.jet, linewidth=0, antialiased=True)
ax.set_title("X Partial Derivative 2")
ax = fig.add_subplot(1, 3, 2, projection='3d')
surf = ax.plot_surface(X, Y, Partial_Der_Y,  cmap=cm.jet, linewidth=0, antialiased=True)
ax.set_title("Y Partial Derivative 2")
ax = fig.add_subplot(1, 3, 3, projection='3d')
surf = ax.plot_surface(X, Y, Laplacian,  cmap=cm.jet, linewidth=0, antialiased=True)
ax.set_title("Laplacian of Gaussian")
plt.show()
```





In [ ]:

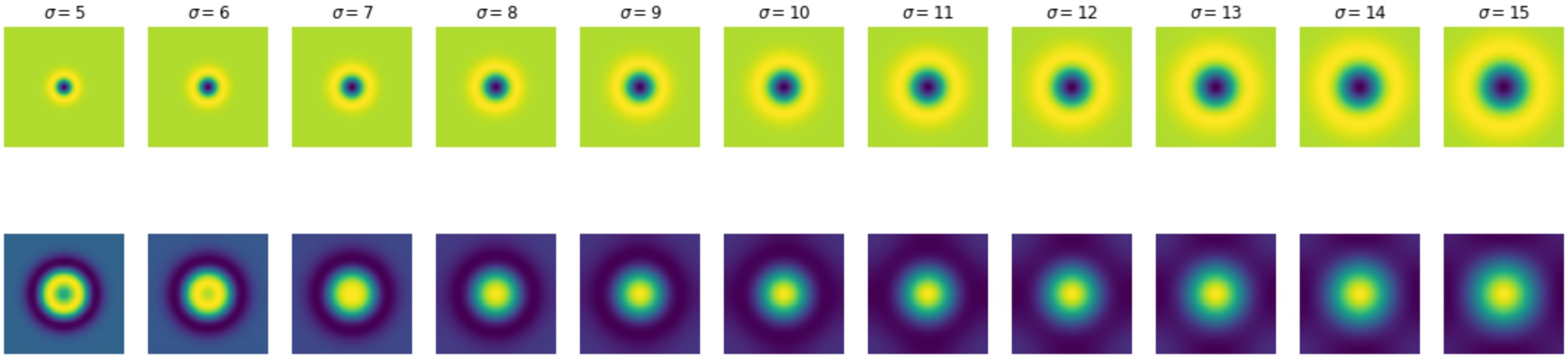
```
w, h = 71,71
hw = w//2
hh = h//2

f = np.ones((h,w), dtype=np.float32)*255
X, Y = np.meshgrid(np.arange(-hh, hh +1, 1), np.arange(-hw, hw +1, 1))

r = w//5
f *= X**2 + Y**2 > r**2

plt.imshow(f)
plt.show()
```



In [ ]:

```
s = 11
fig, ax = plt.subplots(2, s, figsize=(20,5))
scale_space = np.empty((h, w, s), dtype=np.float32)
sigmas = np.arange(5, 16, 1)
for i, sigma in enumerate(np.arange(5, 16, 1)):
    log_hw = 3*np.max(sigmas)
    X_ = np.arange(-log_hw, log_hw + 1, 1)
    Y_ = np.arange(-log_hw, log_hw + 1, 1)
    X, Y = np.meshgrid(X_,Y_)
    Partial_Der_X = np.exp(-(X**2 + Y**2)/(2*sigma**2))*(X**2/sigma**2-1)/(2*m.pi*sigma**4)
    Partial_Der_Y = np.exp(-(X**2 + Y**2)/(2*sigma**2))*(Y**2/sigma**2-1)/(2*m.pi*sigma**4)
    Laplacian = sigma**2*(Partial_Der_X + Partial_Der_Y)
    f_log = cv.filter2D(f, -1, Laplacian)
    scale_space[:,:,i] = f_log
    ax[0, i].imshow(Laplacian)
    ax[0, i].axis('off')
    ax[0, i].set_title(r'$\sigma ={}$'.format(sigma))
    ax[1, i].imshow(f_log)
    ax[1, i].axis('off')

indices = np.unravel_index(np.argmax(scale_space, axis = None), scale_space.shape)
print(indices)
print(sigmas[indices[2]])
```

```
(35, 35, 5)
10
```



In [ ]:

```
Img1 = cv.imread('img1.ppm',cv.IMREAD_COLOR)
Img2 = cv.imread('img2.ppm',cv.IMREAD_COLOR)

assert Img1 is not None
assert Img2 is not None

Gray1 = cv.cvtColor(Img1, cv.COLOR_BGR2RGB)
Gray2 = cv.cvtColor(Img2, cv.COLOR_BGR2RGB)

#keypoints
SIFT = cv.SIFT_create()
Key_Pots_1, Desp_1 = SIFT.detectAndCompute(Img1,None)
Key_Pots_2, Desp_2 = SIFT.detectAndCompute(Img2,None)

BF = cv.BFMatcher(cv.NORM_L1, crossCheck=True)

Match_1 = BF.match(Desp_1,Desp_2)

Match_1 = sorted(Match_1, key = lambda x:x.distance)

New_Img_1 = cv.drawMatches(Gray1, Key_Pots_1, Gray2, Key_Pots_2, Match_1[:150], Gray2, flags=2)

fig,ax = plt.subplots(1,1,figsize=(20,20),sharex='all',sharey='all')
ax.imshow(New_Img_1)
ax.set_title("SIFT Features between Images")
ax.axis('off')
```

Out[ ]:

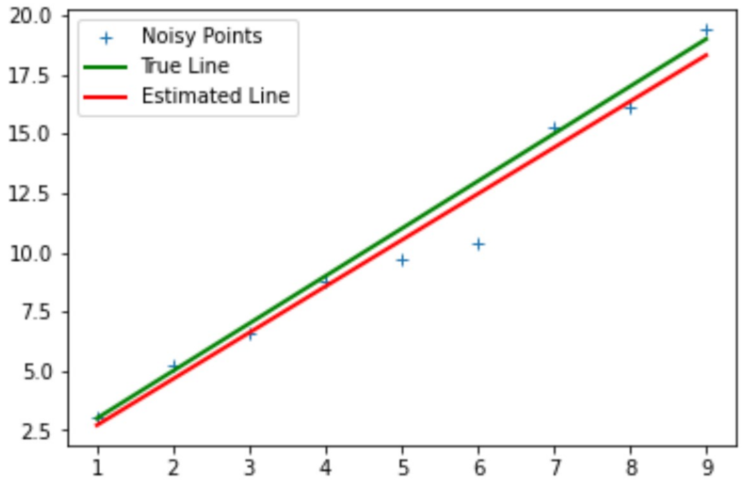
```
(-0.5, 1599.5, 639.5, -0.5)
```

In [ ]:

```
m = 2
c = 1
x = np.arange (1, 10, 1)
np.random.seed(45)
sigma = 1
n = sigma*np.random.randn(len(x))
o = np.zeros(x.shape) # o[-1] = 20 # Outlier
y = m * x + c + n + o

n = len(x)
X = np.concatenate([x.reshape((n,1)), np.ones((n,1))], axis=1)
B = np.linalg.pinv(X.T @ X)@X.T @ y
mstar = B[0]
cstar = B[1]

plt.plot(x, y, '+', label='Noisy Points')
plt.plot([x[0], x[-1]], [m*x[0] + c, m*x[-1] + c], color='g' , linewidth=2, label='True Line')
plt.plot([x[0], x[-1]], [mstar*x[0] + cstar, mstar*x[-1] + cstar], color='r' , linewidth=2, label='Estimated Line')
plt.legend()
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x1c729d75d00>
```

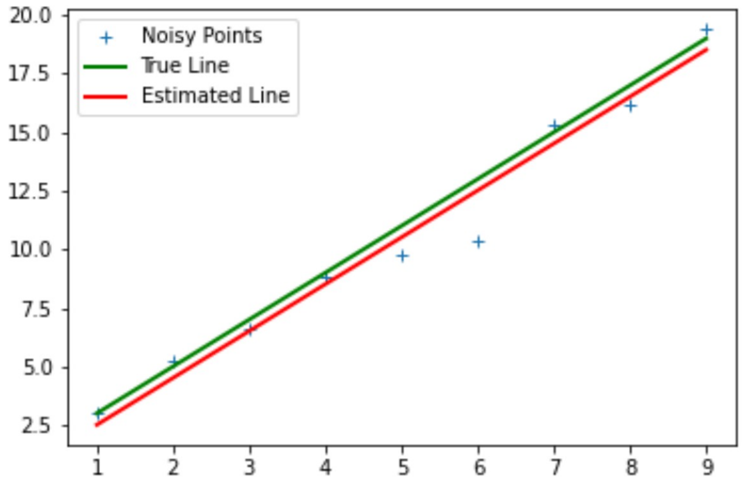


In [ ]:

```
m = 2
c = 1
x = np.arange (1, 10, 1)
np.random.seed(45)
sigma = 1
n = sigma*np.random.randn(len(x))
o = np.zeros(x.shape) # o[-1] = 20 # Outlier
y = m * x + c + n + o

n = len(x)

u11 = np.sum((x - np.mean(x))**2 )
u12 = np.sum((x - np.mean(x))*(y - np.mean(y)))
u21 = u12
u22 = np.sum((y - np.mean(y))**2 )

U = np.array([[u11,u12], [u21,u22]])

W, V = np.linalg.eig(U)
Eig_Vec_to_smallest_ev = V[:, np.argmin(W)]

a = Eig_Vec_to_smallest_ev[0]
b = Eig_Vec_to_smallest_ev[1]
d = a*np.mean(x) + b*np.mean(y)

mstar = -a/b
cstar = d/b

plt.plot(x, y, '+', label='Noisy Points')
plt.plot([x[0], x[-1]], [m*x[0] + c, m*x[-1] + c], color='g' , linewidth=2, label='True Line')
plt.plot([x[0], x[-1]], [mstar*x[0] + cstar, mstar*x[-1] + cstar], color='r' , linewidth=2, label='Estimated Line')
plt.legend()
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x291ff82fe80>
```


+ Noisy Points
True Line
Estimated Line