Name : Chathumini B.G.D.T.

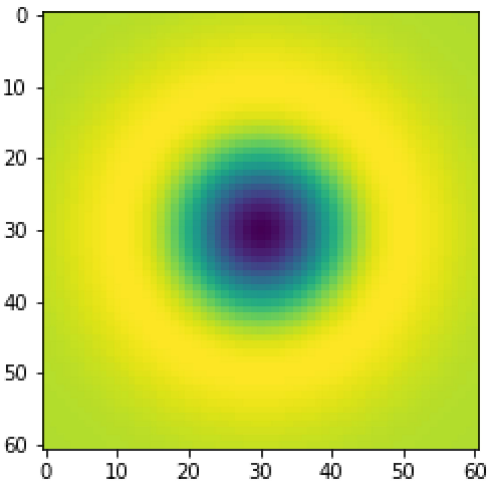
Index Number : 190107T

Github Repo: https://github.com/dulmi-19/Image-Processing-and-Machine-Vision

Question 1

In [ ]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm

sigma =10
hw = 3*sigma
X,Y = np.meshgrid(np.arange(-hw,hw + 1, 1), np.arange(-hw,hw + 1, 1))
log = 1/(2*np.pi*sigma**2)*(X**2/(sigma**2)+Y**2/(sigma**2)-2)*np.exp(-(X**2+Y**2),

plt.imshow(log)
```

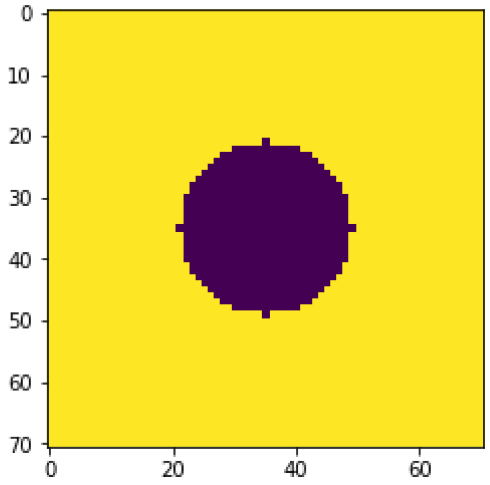
Out[ ]:
```
<matplotlib.image.AxesImage at 0x1b60d390760>
```



Question 2

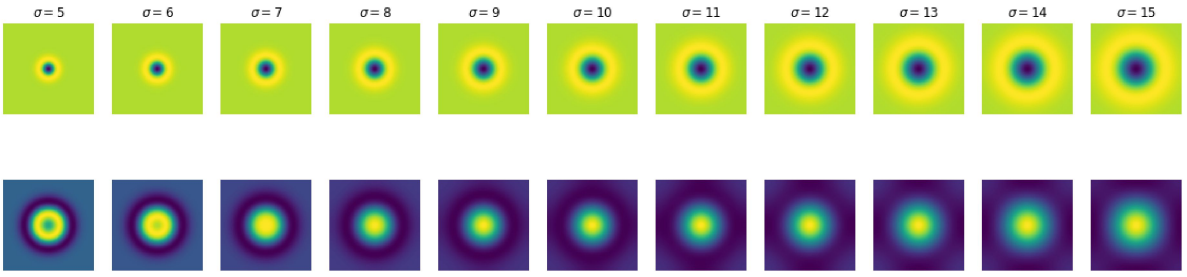
In [ ]:
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

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

Out[ ]:
```
<matplotlib.image.AxesImage at 0x1b60f499690>
```



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(sigmas):
    log_hw =3*np.max(sigmas)
    X,Y = np.meshgrid(np.arange(-log_hw,log_hw + 1, 1), np.arange(-log_hw,log_hw +
    log = 1/(2*np.pi*sigma**2)*(X**2/(sigma**2)+Y**2/(sigma**2)-2)*np.exp(-(X**2+Y
    f_log = cv.filter2D(f,-1,log)
    scale_space[:,:,i]=f_log
    ax[0,i].imshow(log)
    ax[0,i].axis('off')
    ax[0,i].set_title('$\sigma = {}$'.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) #sqrt{2}*sigma
print(sigmas[indices[2]])
```

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



Justification- Since radious of the blob is 14,scale-space extremum is when σ =10 , $(r/\sqrt{2})$

Question 3

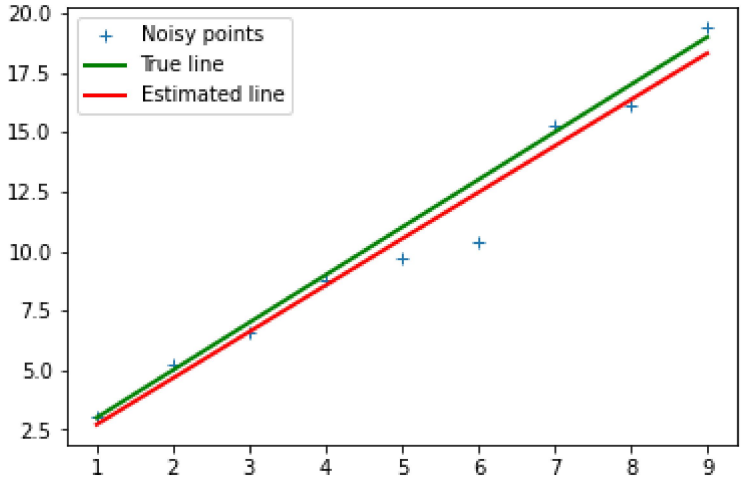
In [ ]:
```
import cv2 as cv
import matplotlib.pyplot as plt
%matplotlib inline

graf_1 = cv.imread('img3.ppm')
graf_2 = cv.imread('img4.ppm')

graf_1 = cv.cvtColor(graf_1, cv.COLOR_BGR2GRAY)
graf_2 = cv.cvtColor(graf_2, cv.COLOR_BGR2GRAY)

#sift
sift = cv.SIFT_create()
```

```
keypoints_1, descriptors_1 = sift.detectAndCompute(graf_1,None)
keypoints_2, descriptors_2 = sift.detectAndCompute(graf_2,None)

#feature matching
bf = cv.BFMatcher(cv.NORM_L1, crossCheck=True)

matches = bf.match(descriptors_1,descriptors_2)
matches = sorted(matches, key = lambda x:x.distance)

plt.figure(figsize=(20,10))
image= cv.drawMatches(graf_1, keypoints_1, graf_2, keypoints_2, matches[:50], graf_
plt.imshow(image)
plt.xticks([]),plt.yticks([])
plt.show()
```

Question 4

In [ ]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
m = 2 # Line equation : y = m*x + c . m i s the s lope . c i s the int e r c ept .
c = 1
x = np.arange(1 ,10 , 1)
np.random.seed(45)
noise = np.random.randn(len(x))
o = np.zeros(x.shape)
# o[=1] = 20
y = m*x + c + noise + 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=r'True
plt.plot([x[0],x[-1]], [mstar*x[0]+cstar, mstar*x[-1]+cstar],color ='r',linewidth =
plt.legend()
```

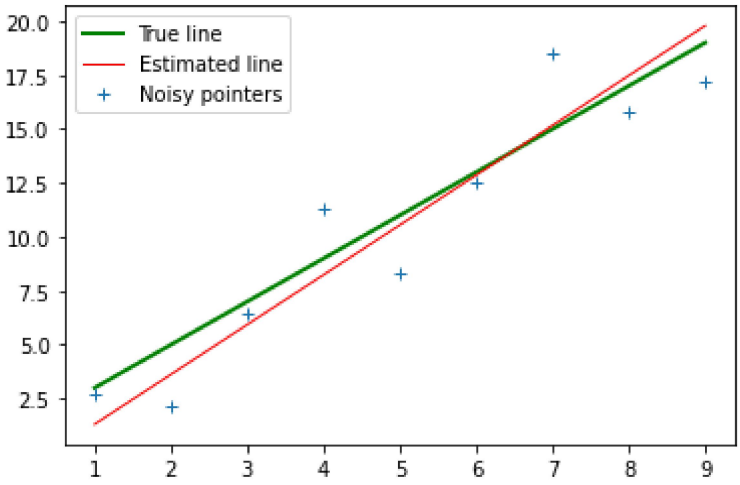
Out[ ]:
```
<matplotlib.legend.Legend at 0x1b61178acb0>
```



Question 5

In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline

m = 2 # Line equation : y = m*x + c . m i s the s lope . c i s the int e r c ep t
c = 1
x = np.arange(1 ,10 , 1)
noise = 2.*np.random.randn(len(x))
o = np.zeros(x.shape)
# o[=1] = 20
y = m*x + c + noise + 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)
ev_corresponding_to_smallest_ev=V[:,np.argmin(W)]

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

mstar=-a/b
cstar=d/b

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


True line
Estimated line
Noisy pointers
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
1
2
3
4
5
6
7
8
9