

Name : Dilshan J.V.A.P

Index number : 190144D

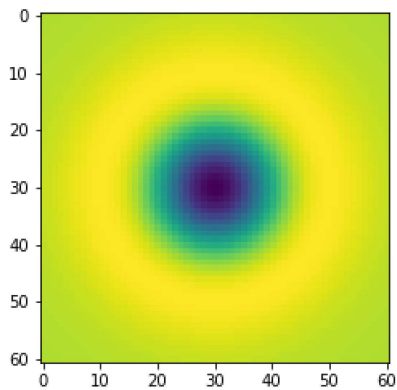
## Exercise 5

```
In [ ]: #Q1
import numpy as np
import matplotlib.pyplot as plt

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)/(2*sigma**2))
plt.imshow(log)
```

```
Out[ ]: <matplotlib.image.AxesImage at 0x18ab64b2970>
```

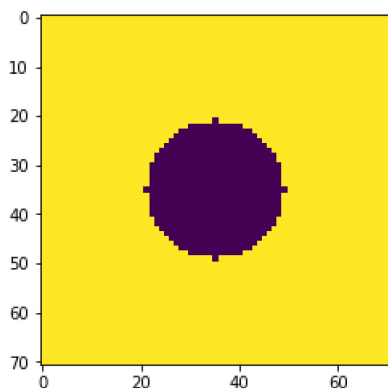


```
In [ ]: #Q2
#generating the circle
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

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 #14
f*=X**2+Y**2>r**2
plt.imshow(f)
```

```
Out[ ]: <matplotlib.image.AxesImage at 0x18ab653f220>
```



```
In [ ]: s=11
fig,ax=plt.subplots(2,s,figsize=(20,5))
```

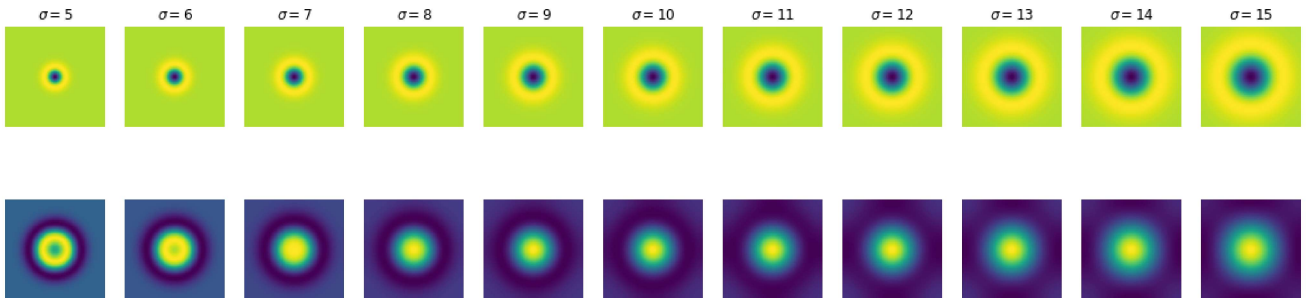
```

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+1,1))
    log=1/(2*np.pi*sigma**2)*(X**2/(sigma**2)+Y**2/(sigma**2)-2)*np.exp(-(X**2+Y**2)/(2*sigma**2))
    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(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(sigmas[indices[2]])

```

10



Scalar space extremum get to a maximum when  $r = 2^{0.5}\sigma$ . We take  $r$  as 14 initially. Therefore the maximum scalar space must give when  $\sigma = 14/(2^{0.5}) = 9.89 \implies 10$

In [ ]:

```

#Q3
img1 = cv.imread('img1.ppm')
img2 = cv.imread('img2.ppm')
img1 = cv.cvtColor(img1, cv.COLOR_BGR2GRAY)
img2 = cv.cvtColor(img2, cv.COLOR_BGR2GRAY)
sift = cv.SIFT_create()
keypoints_1, descriptors_1 = sift.detectAndCompute(img1, None)
keypoints_2, descriptors_2 = sift.detectAndCompute(img2, None)
bf = cv.BFMatcher(cv.NORM_L1, crossCheck=True)
matches = bf.match(descriptors_1, descriptors_2)
matches = sorted(matches, key = lambda x:x.distance)
img3 = cv.drawMatches(img1, keypoints_1, img2, keypoints_2, matches[:50], img2, flags=2)
plt.figure(figsize=(15,15))
plt.imshow(img3)
plt.xticks([], plt.yticks([]))
plt.show()

```



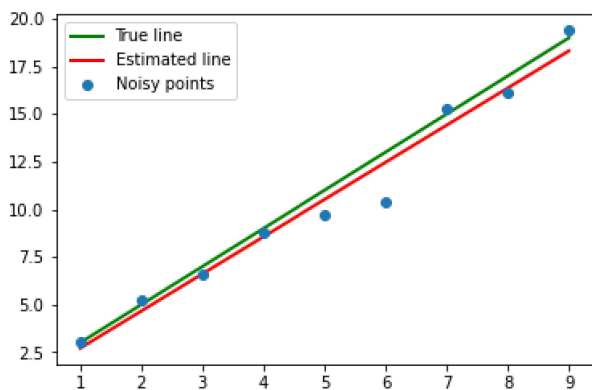
In [ ]:

```
#Q4
# Least-squares line fitting
m = 2 #gradient
c = 1 #intercept
x = np.arange(1,10,1)
np.random.seed(45)
noise=np.random.randn(len(x))
n = 2.*np.random.randn(len(x))
o = np.zeros(x.shape)
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[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=2,label=r'Estimated line')
plt.plot(x,y,'o',label='Noisy points')
plt.legend()
```

Out[ ]: <matplotlib.legend.Legend at 0x18ab5decfa0>



In [ ]:

```
#Q5
# Least-squares line fitting
m = 2 #gradient
c = 1 #intercept
x = np.arange(1,10,1)
np.random.seed(45)
noise=np.random.randn(len(x))
n = 2.*np.random.randn(len(x))
o = np.zeros(x.shape)
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], [u12,u22]])
W,V = np.linalg.eig(U)
ev_corresponding_to_smalest_ev = V[:,np.argmin(W)]

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

mstar = -a/b
cstar=d/b

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[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]], [m*x[0]+cstar, mstar*x[-1]+cstar], color='r', linewidth = 2, label=r'Estimated line')
plt.plot(x,y,'+',label = "Noisy points")
plt.legend(loc="best")
plt.show()
```

