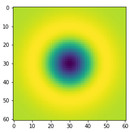
Excercise -05

190539T

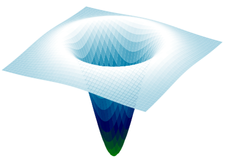
Q1>

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

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 0x26a4ac9d400>
```



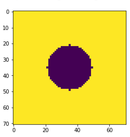
In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter

fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(111, projection='3d')
surf = ax.plot_surface(x, y, log, cmap=cm.ocean, linewidth = 0, antialiased=True)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))

plt.axis("off")
plt.show()
```



Q2>

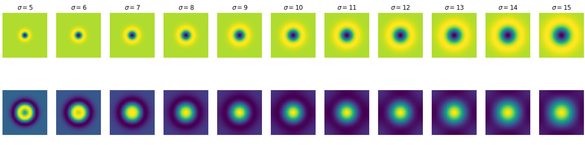
In [ ]:
```
w,h=71,71
hw=w//2
hh=h//2

f=np.ones((h,w),dtype=np.float32)*255
x,y=np.meshgrid(np.arange(-hw,hw+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 0x26a4acfc250>
```



In [ ]:
```
s= 11

fix,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,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(indices)
print(sigmas[indices[2]])
```

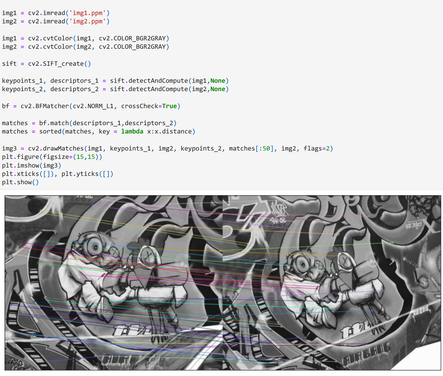
```
(35, 35, 5)
10
```






Q3>

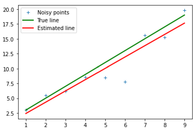
In [ ]:
```
import cv2
import matplotlib.pyplot as plt

img1 = cv2.imread('img1.ppm')
img2 = cv2.imread('img2.ppm')

img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2GRAY)
img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2GRAY)

sift = cv2.SIFT_create()

keypoints_1, descriptors_1 = sift.detectAndCompute(img1,None)
keypoints_2, descriptors_2 = sift.detectAndCompute(img2,None)

bf = cv2.BFMatcher(cv2.NORM_L1, crossCheck=True)

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

img3 = cv2.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()
```



Q4>

In [ ]:
```
m = 2 # Line equation : y = m*x + c . m is the slope . c is the intercept .
c = 1
x = np . arange (1 ,10 , 1)
n = 2.*np .random. randn ( len ( x ) )
np.random.seed(45)
o = np . zeros ( x . shape )
# o[-1] = 20
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 0x26a4b4c0490>
```




Q5>

In [ ]:
```
m = 2 # Line equation : y = m*x + c . m is the slope . c is the intercept .
c = 1
x = np . arange (1 ,10 , 1)
n = 2.*np .random. randn ( len ( x ) )
np.random.seed(45)
o = np . zeros ( x . shape )
# o[-1] = 20
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)
ev_cor_to_small_ev= V[:,np.argmin(W)]

a=ev_cor_to_small_ev[0]
b=ev_cor_to_small_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=1,label='Estimated line')

plt.legend(loc='best')
```

Out[ ]:
```
<matplotlib.legend.Legend at 0x26a4cba37f0>
```