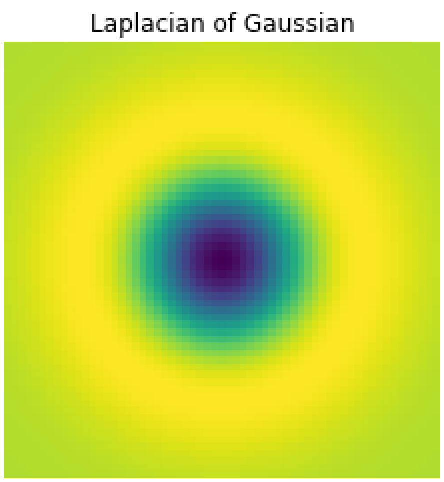
Name: Sumanasekara W.K.G.G.

Index: 190610E

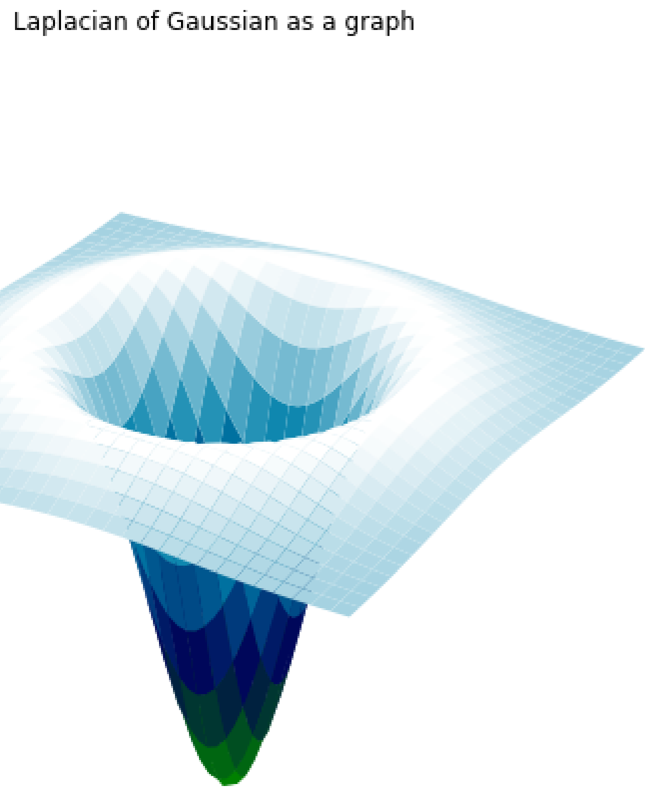
- Blobs

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

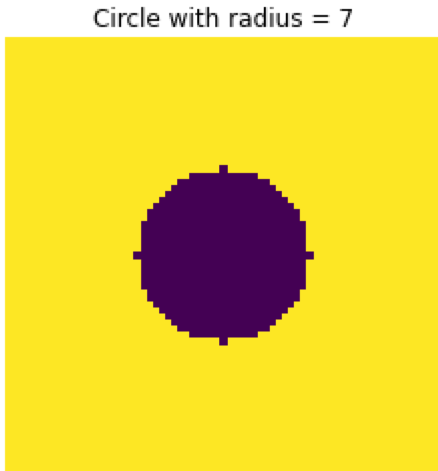
In [ ]:
```
sigma = 10
hw = 3*sigma

XX, YY = np.meshgrid(np.arange(-hw, hw+1, 1), np.arange(-hw, hw+1, 1))
LoG = 1/(2*np.pi*sigma**2)*(XX**2/(sigma**2) + YY**2/(sigma**2) -2)*np.exp(-(XX**2 + YY

plt.imshow(LoG)
plt.title("Laplacian of Gaussian")
plt.axis("off")
plt.show()
```



In [ ]:
```
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111, projection = '3d')

surf = ax.plot_surface(XX, YY, LoG, cmap=cm.ocean, linewidth=0, antialiased=True)
ax.zaxis.set_major_locator(LinearLocator(10))
plt.title("Laplacian of Gaussian as a graph")
plt.axis("off")
plt.show()
```

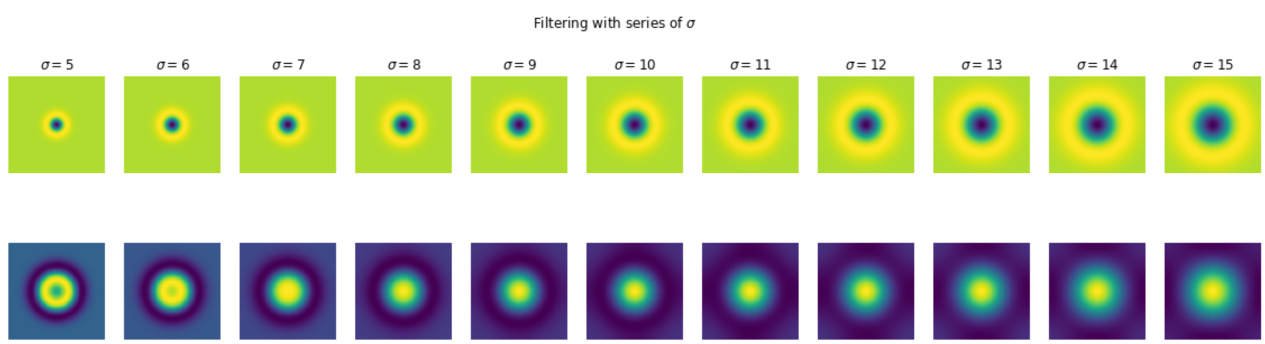


In [ ]:
```
h, w = 71, 71
hh, hw = h//2, w//2
radius = w//5

circle = np.ones((h, w), dtype=np.float32)*255
XX, YY = np.meshgrid(np.arange(-hh, hh+1, 1), np.arange(-hw, hw+1, 1))
circle *= XX**2+YY**2 > radius**2

plt.imshow(circle)
plt.axis("off")
plt.title("Circle with radius = 7")
plt.show()
```



In [ ]:

```
sigmas = np.arange(5, 16, 1)
sigma_count = len(sigmas)
scale_space = np.empty((h, w, sigma_count), dtype=np.float32)
log_wh = 3*np.max(sigmas)
XX, YY = np.meshgrid(np.arange(-log_wh, log_wh+1, 1), np.arange(-log_wh, log_wh+1, 1))
fig, ax = plt.subplots(2, sigma_count, figsize=(20, 5))

for i, sigma in enumerate(sigmas):
    LoG = 1/(2*np.pi*sigma**2)*(XX**2/(sigma**2) + YY**2/(sigma**2) -2)*np.exp(-(XX**2
    circle_filtered = cv.filter2D(circle, -1, LoG)
    scale_space[:, :, i] = circle_filtered

    ax[0, i].set_title("$\sigma = {}$".format(sigma))
    ax[0, i].imshow(LoG)
    ax[0, i].axis("off")
    ax[1, i].imshow(circle_filtered)
    ax[1, i].axis("off")

fig.suptitle("Filtering with series of $\sigma$")
plt.show()
```



In [ ]:

```
indices = np.unravel_index(np.argmax(scale_space, axis=None), scale_space.shape)
print("scale-space extremum = {}".format(sigmas[indices[2]]))
```

```
scale-space extremum = 10
```

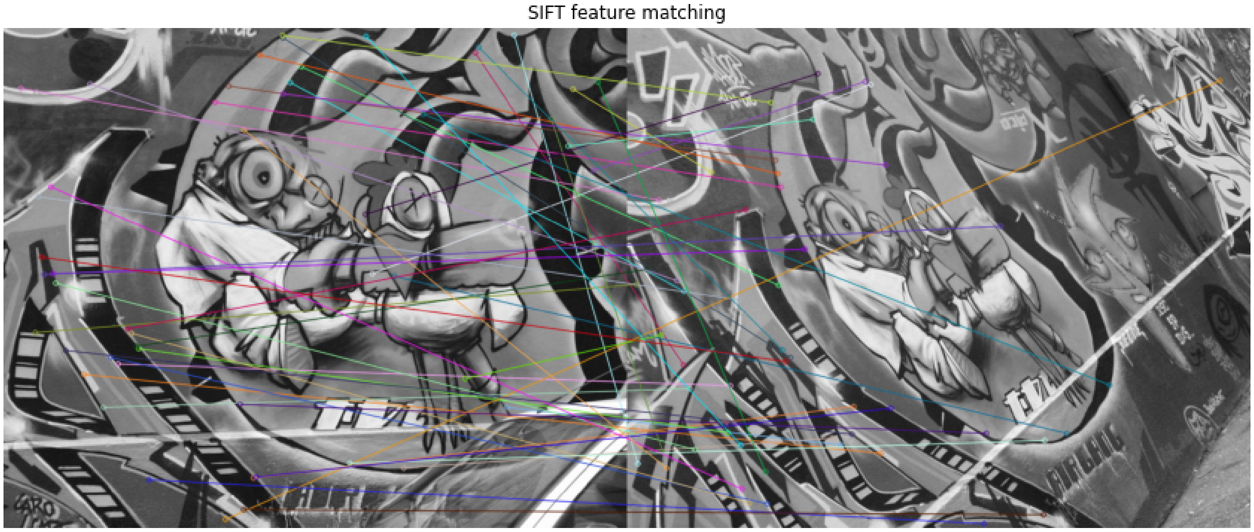
Calculated $\sigma = \frac{14}{\sqrt{2}} \approx 10$

In [ ]:

```
img1 = cv.imread("graffiti\img1.ppm")
img2 = cv.imread("graffiti\img4.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

fig, ax = plt.subplots(figsize=(15, 12))
ax.imshow(img3)
plt.axis('off')
plt.title("SIFT feature matching")
plt.show()
```

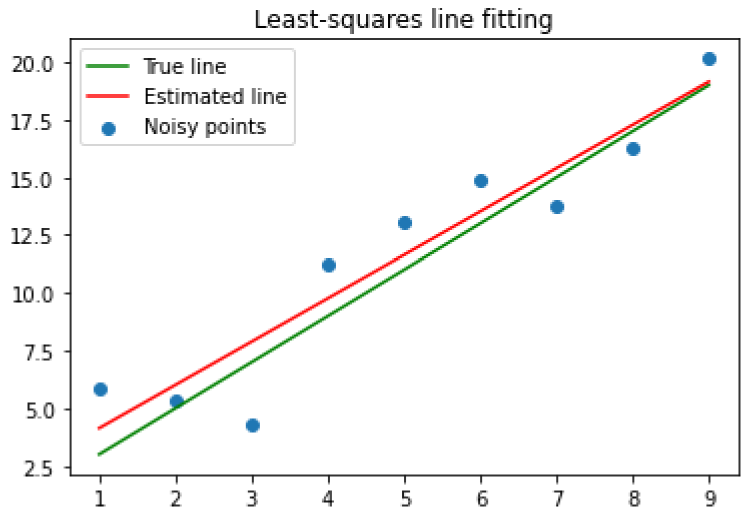
SIFT feature matching



- Fitting

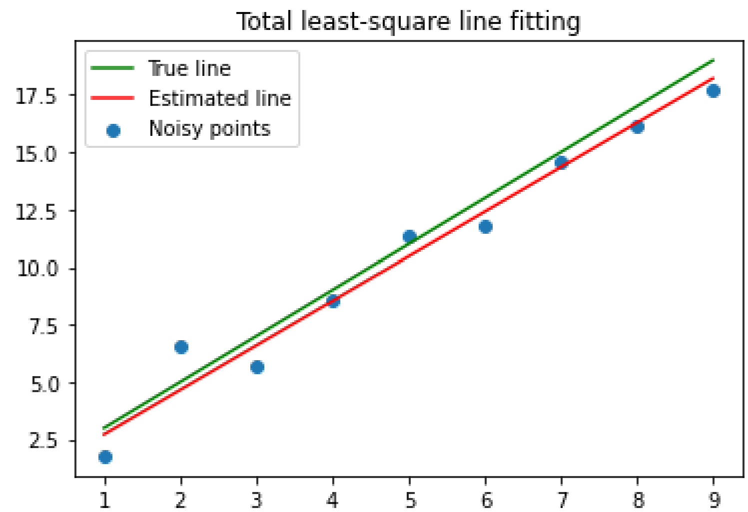
In [ ]:

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

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

plt.plot((x[0], x[-1]), (m*x[0]+c, m*x[-1]+c), 'g', label="True line")
plt.plot((x[0], x[-1]), (mstar*x[0]+cstar, mstar*x[-1]+cstar), 'r', label="Estimated li
plt.scatter(x, y, label='Noisy points')
plt.legend(loc='best')
plt.title("Least-squares line fitting")
plt.show()
```



In [ ]:

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

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_of_smallest_ef = V[:, np.argmin(W)]
a, b = ev_of_smallest_ef[0], ev_of_smallest_ef[1]
d = a*np.mean(x) + b*np.mean(y)

mstar, cstar = -a/b, d/b

plt.plot((x[0], x[-1]), (m*x[0]+c, m*x[-1]+c), 'g', label="True line")
plt.plot((x[0], x[-1]), (mstar*x[0]+cstar, mstar*x[-1]+cstar), 'r', label="Estimated li
plt.scatter(x, y, label='Noisy points')
plt.legend(loc='best')
plt.title("Total least-square line fitting")
plt.show()
```


Total least-square line fitting
True line
Estimated line
Noisy points
17.5
15.0
12.5
10.0
7.5
5.0
2.5
1
2
3
4
5
6
7
8
9