**Department of Computer Science and Engineering (Data Science)**
**Academic Year 2022-2023**

**Name : Sarvagya Singh** **SAPID : 60009200030** **Batch : K1**

**AIM: To implement Ideal Low Pass & High Pass Filtering on an image**

**THEORY:**

**1. Ideal Low Pass Filter**

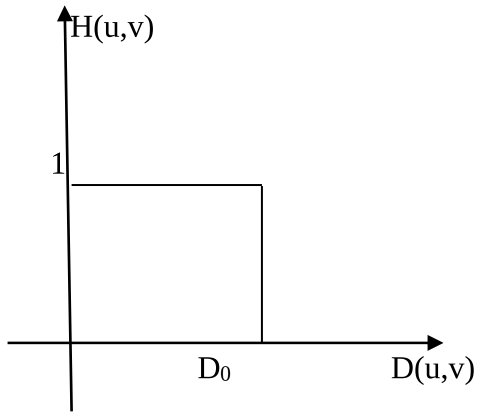
This filter cuts off all high frequency components of the Fourier transform that are at a distance greater than a specified distance $D_0$.

$$H(u,v) = 1; \text{ if } D(u,v) < D_0$$

$$= 0; \text{ if } D(u,v) > D_0$$

Where,

$D_0$ is the specified non negative distance.




Response of Ideal Low Pass Filter

D(u,v) is the distance from the point (u,v) to the origin of the frequency rectangle for an M X N image.

$$D(u,v)=[(u-M/2)^2 + (v-N/2)^2]^{\wedge}\, ½$$

Therefore ,

For an image, when u=M/2 , v=N/2

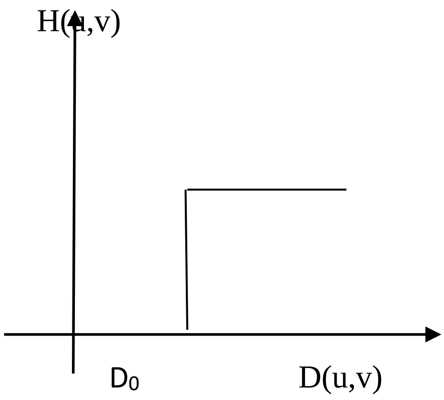
$$D(u,v)=0$$

This formula centers our H(u,v).

D(u,v) gives us concentric rings with each ring having a fixed value.

**2. Ideal High Pass Filter**

This filter cuts off all high frequency components of the Fourier transform that are at a distance greater than a specified distance $D_0$.




$$H(u,v) = 0; \text{ if } D(u,v) < D0$$

$$= 1; \text{ if } D(u,v) > D0$$

Where,

D0 is the specified non negative distance.

D(u,v) is the distance from the point (u,v) to the origin of the frequency rectangle for an M X N image.

**RESULT:**

An ideal low-pass filter completely eliminates all frequencies above the cutoff frequency while passing those below unchanged.Ideal low pass filters blurs the image where the High pass filters only shows/highlights the part of the image which is important

Ideal low pass filter and High pass filter

Sarvayga Singh -- 60009200030 (K1)

Lab8

In [1]:
```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
import matplotlib.pyplot as plt
```

In [2]:
```
def step(num):
  num_list = num.split('.')
  if num_list[1] > 5:
    return int(num_list(1))+1
  else:
    return int(num_list[0])
```

In [ ]:
```
img = cv2.imread('/content/image.jpg',0)
cv2_imshow(img)
```

In [ ]:
```
M,N = img.shape[0],img.shape[1]
M,N
```

Out[ ]:
```
(564, 849)
```

In [ ]:
```
np.max(img),np.min(img)
```

Out[ ]:
```
(218, 169)
```

In [ ]:
```
def dist(u,v):
  return np.sqrt((u - M/2)**2+(v-N/2)**2)
```

In [ ]:
```
F = np.fft.fft2(img)
Fshift = np.fft.fftshift(F)
```

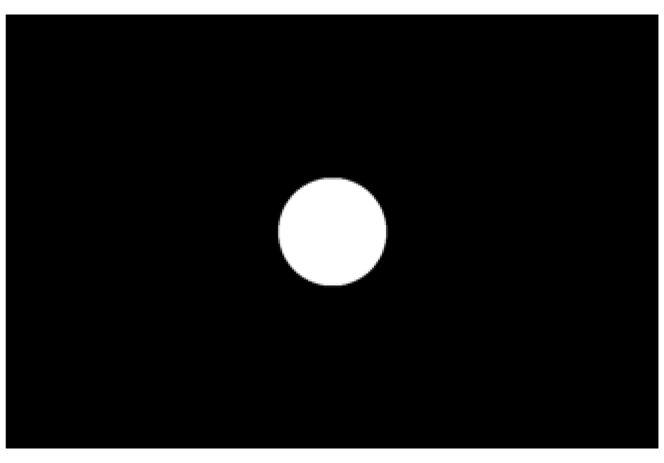
In [ ]:
```
h = [[0 for i in range(N)] for j in range(M)]
h = np.array(h)
threshold = 70
print(len(h[0]),len(h))
```

```
849 564
```

In [ ]:
```
for i in range(0,M): #y axis
  for j in range(0,N): # x axis
    d = dist(i,j)
    if d<=threshold:
      h[i,j] = 1
    else:
      h[i,j] = 0
```

In [ ]:
```
plt.imshow(h, cmap='gray')
plt.axis('off')
plt.show()

Gshift = Fshift * h
G = np.fft.ifftshift(Gshift)
g = np.abs(np.fft.ifft2(G))
plt.imshow(g, cmap='gray')
plt.axis('off')
plt.show()
```

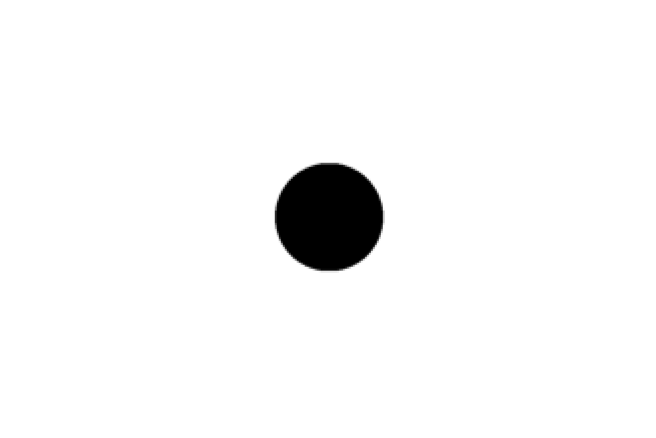


In [ ]:
```
np.shape(h)
```

Out[ ]:
```
(564, 849)
```

In [ ]:
```
i = [[1 for i in range(N)] for j in range(M)]
```

In [ ]:
```
# cv2_imshow(np.array(h))
# Filter: High pass filter
h = 1 - h
plt.imshow(h, cmap='gray')
plt.axis('off')
plt.show()
Gshift = Fshift * h
G = np.fft.ifftshift(Gshift)
g = np.abs(np.fft.ifft2(G))
plt.imshow(g, cmap='gray')
plt.axis('off')
plt.show()
```



In [ ]:
```
# calculating the discrete Fourier transform
DFT = cv2.dft(np.float32(img), flags=cv2.DFT_COMPLEX_OUTPUT)

# reposition the zero-frequency component to the spectrum's middle
shift = np.fft.fftshift(DFT)
row, col = img.shape
center_row, center_col = row // 2, col // 2

# create a mask with a centered square of 1s
mask = np.zeros((row, col, 2), np.uint8)
mask[center_row - 30:center_row + 30, center_col - 30:center_col + 30] = 1

# put the mask and inverse DFT in place.
fft_shift = shift * mask
fft_ifft_shift = np.fft.ifftshift(fft_shift)
imageThen = cv2.idft(fft_ifft_shift)

# calculate the magnitude of the inverse DFT
imageThen = cv2.magnitude(imageThen[:,:,0], imageThen[:,:,1])

# visualize the original image and the magnitude spectrum
plt.figure(figsize=(10,10))
plt.subplot(121), plt.imshow(img, cmap='gray')
plt.title('Input Image'), plt.xticks([]), plt.yticks([])
plt.subplot(122), plt.imshow(imageThen, cmap='gray')
plt.title('Low pass filter'), plt.xticks([]), plt.yticks([])
plt.show()
```


In [ ]: