Visión Computacional para imágenes y video (Gpo 10)

Alumnos:

- Armando Bringas Corpus - A01200230
- Guillermo Alfonso Muñiz Hermosillo - A01793101
- Jorge Luis Arroyo Chavelas - A01793023
- Samantha R Mancias Carrillo - A01196762
- Sofia E Mancias Carrillo - A01196563

Profesores:

- Dr. Gilberto Ochoa Ruiz
- Mtra. Yetnalezi Quintas Ruiz

5. Frequency Domain

Table of Contents

1. Libraries
2. Fast Fourier Transform (FFT)
3. Low Pass Filter
4. High Pass Filter
5. Excercises

   a. Low Pass Filter application

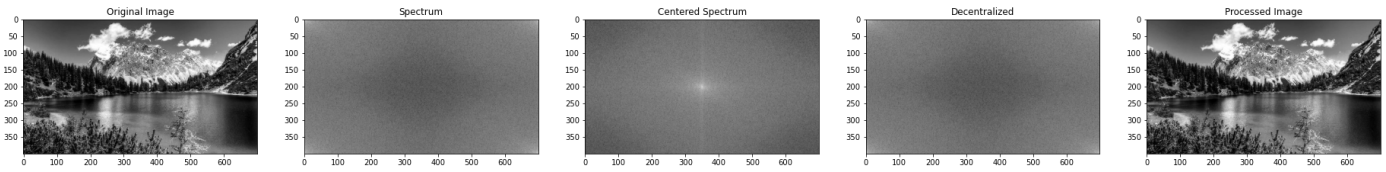
   b. High Pass Filter application

Importing Libraries

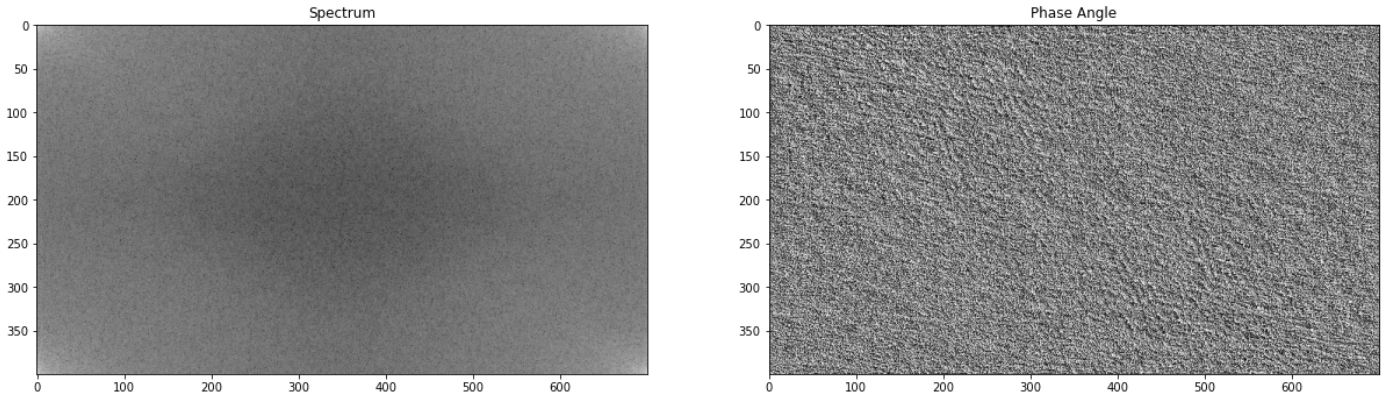
In [1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from math import sqrt,exp
```

Fast Fourier Transform (FFT)

In [2]:
```
plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

img = cv2.imread("data/image.jpg", 0)
plt.subplot(151), plt.imshow(img, "gray"), plt.title("Original Image")

original = np.fft.fft2(img)
plt.subplot(152), plt.imshow(np.log(1+np.abs(original)), "gray"), plt.title("Spectrum")

center = np.fft.fftshift(original)
plt.subplot(153), plt.imshow(np.log(1+np.abs(center)), "gray"), plt.title("Centered Spec

inv_center = np.fft.ifftshift(center)
plt.subplot(154), plt.imshow(np.log(1+np.abs(inv_center)), "gray"), plt.title("Decentral

processed_img = np.fft.ifft2(inv_center)
plt.subplot(155), plt.imshow(np.abs(processed_img), "gray"), plt.title("Processed Image"

plt.show()
```



In [3]:
```
plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

img = cv2.imread("data/image.jpg", 0)

original = np.fft.fft2(img)
plt.subplot(131), plt.imshow(np.log(np.abs(original)), "gray"), plt.title("Spectrum")

plt.subplot(132), plt.imshow(np.angle(original), "gray"), plt.title("Phase Angle")
plt.show()
```



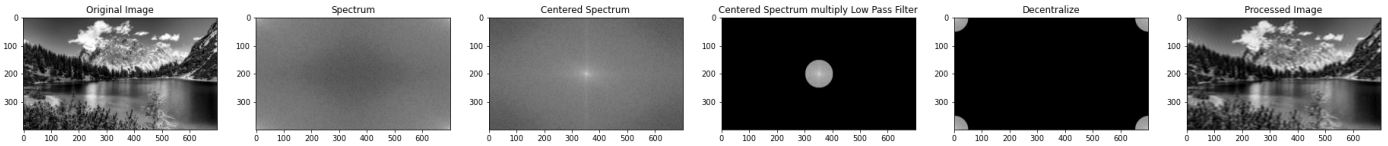
In [4]:
```
def distance(point1,point2):
    return sqrt((point1[0]-point2[0])**2 + (point1[1]-point2[1])**2)

def idealFilterLP(D0,imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            if distance((y,x),center) < D0:
                base[y,x] = 1
    return base

def idealFilterHP(D0,imgShape):
```

```
    base = np.ones(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            if distance((y,x),center) < D0:
                base[y,x] = 0
    return base

def butterworthLP(D0,imgShape,n):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = 1/(1+(distance((y,x),center)/D0)**(2*n))
    return base

def butterworthHP(D0,imgShape,n):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = 1-1/(1+(distance((y,x),center)/D0)**(2*n))
    return base

def gaussianLP(D0,imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = exp(((-distance((y,x),center)**2)/(2*(D0**2))))
    return base

def gaussianHP(D0,imgShape):
    base = np.zeros(imgShape[:2])
    rows, cols = imgShape[:2]
    center = (rows/2,cols/2)
    for x in range(cols):
        for y in range(rows):
            base[y,x] = 1 - exp(((-distance((y,x),center)**2)/(2*(D0**2))))
    return base
```

In [5]:

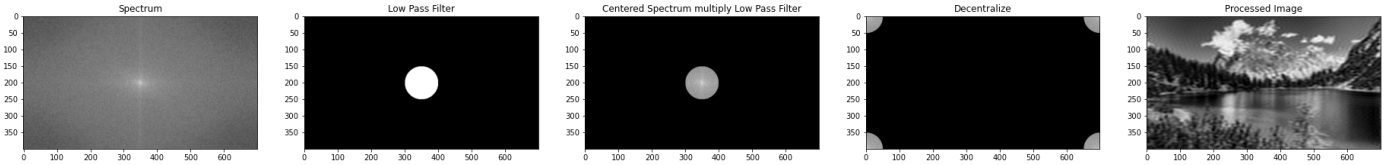
```
plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

img = cv2.imread("data/image.jpg", 0)
plt.subplot(161), plt.imshow(img, "gray"), plt.title("Original Image")

original = np.fft.fft2(img)
plt.subplot(162), plt.imshow(np.log(1+np.abs(original)), "gray"), plt.title("Spectrum")

center = np.fft.fftshift(original)
plt.subplot(163), plt.imshow(np.log(1+np.abs(center)), "gray"), plt.title("Centered Spec

LowPassCenter = center * idealFilterLP(50,img.shape)
plt.subplot(164), plt.imshow(np.log(1+np.abs(LowPassCenter)), "gray"), plt.title("Center

LowPass = np.fft.ifftshift(LowPassCenter)
plt.subplot(165), plt.imshow(np.log(1+np.abs(LowPass)), "gray"), plt.title("Decentralize

inverse_LowPass = np.fft.ifft2(LowPass)
plt.subplot(166), plt.imshow(np.abs(inverse_LowPass), "gray"), plt.title("Processed Imag
```

```
plt.show()
```



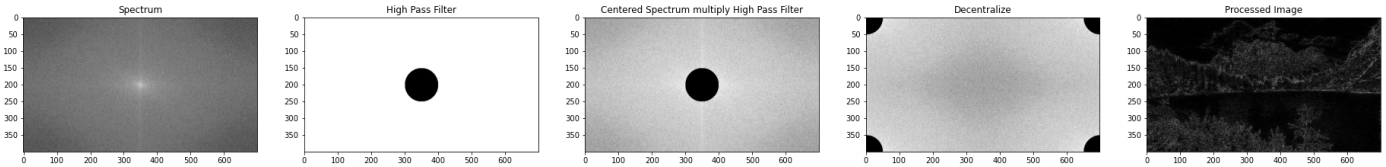
Low Pass Filter

In [6]:
```
img = cv2.imread("data/image.jpg", 0)
original = np.fft.fft2(img)
center = np.fft.fftshift(original)

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

plt.subplot(151), plt.imshow(np.log(1+np.abs(center)), "gray"), plt.title("Spectrum")

LowPass = idealFilterLP(50,img.shape)
plt.subplot(152), plt.imshow(np.abs(LowPass), "gray"), plt.title("Low Pass Filter")

LowPassCenter = center * idealFilterLP(50,img.shape)
plt.subplot(153), plt.imshow(np.log(1+np.abs(LowPassCenter)), "gray"), plt.title("Center

LowPass = np.fft.ifftshift(LowPassCenter)
plt.subplot(154), plt.imshow(np.log(1+np.abs(LowPass)), "gray"), plt.title("Decentralize

inverse_LowPass = np.fft.ifft2(LowPass)
plt.subplot(155), plt.imshow(np.abs(inverse_LowPass), "gray"), plt.title("Processed Imag

plt.show()
```

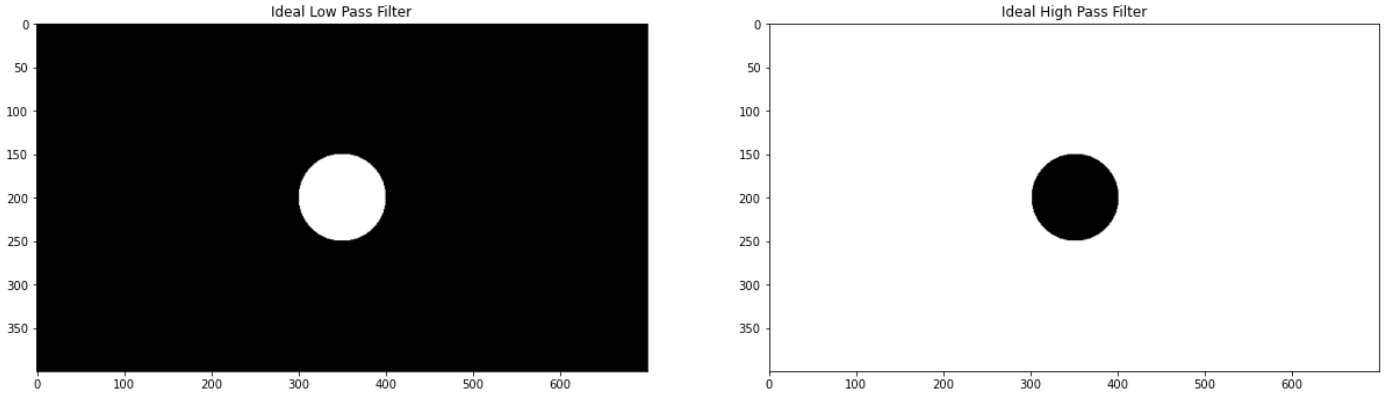


High Pass Filter

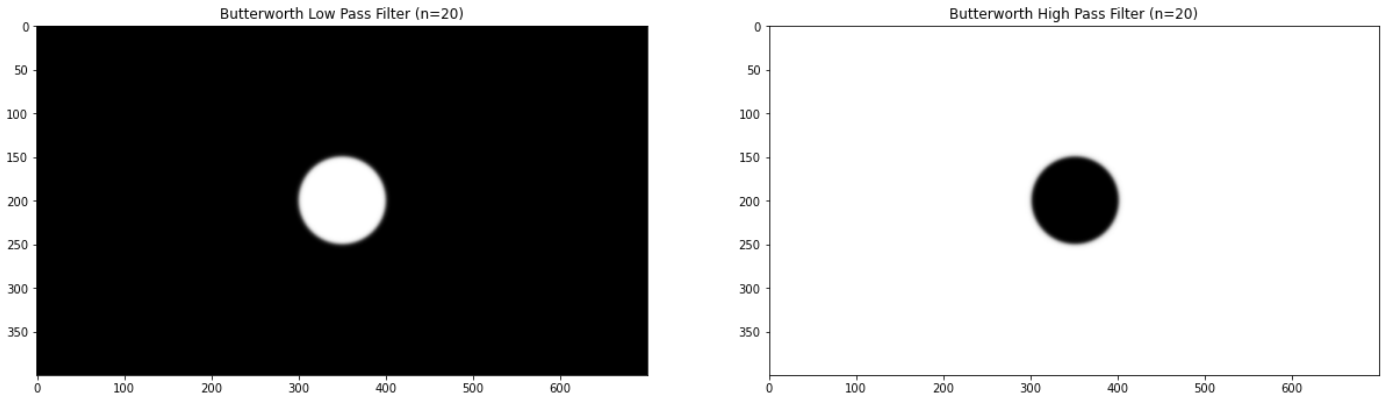
In [7]:
```
img = cv2.imread("data/image.jpg", 0)
original = np.fft.fft2(img)
center = np.fft.fftshift(original)

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

plt.subplot(151), plt.imshow(np.log(1+np.abs(center)), "gray"), plt.title("Spectrum")

HighPass = idealFilterHP(50,img.shape)
plt.subplot(152), plt.imshow(np.abs(HighPass), "gray"), plt.title("High Pass Filter")

HighPassCenter = center * idealFilterHP(50,img.shape)
plt.subplot(153), plt.imshow(np.log(1+np.abs(HighPassCenter)), "gray"), plt.title("Cente

HighPass = np.fft.ifftshift(HighPassCenter)
plt.subplot(154), plt.imshow(np.log(1+np.abs(HighPass)), "gray"), plt.title("Decentraliz

inverse_HighPass = np.fft.ifft2(HighPass)
plt.subplot(155), plt.imshow(np.abs(inverse_HighPass), "gray"), plt.title("Processed Ima

plt.show()
```



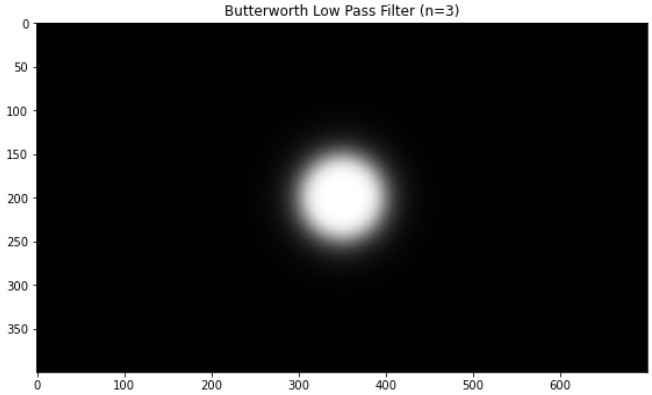
In [8]:
```
plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

LowPass = idealFilterLP(50,img.shape)
plt.subplot(131), plt.imshow(LowPass, "gray"), plt.title("Ideal Low Pass Filter")

HighPass = idealFilterHP(50,img.shape)
plt.subplot(132), plt.imshow(HighPass, "gray"), plt.title("Ideal High Pass Filter")

plt.show()
```

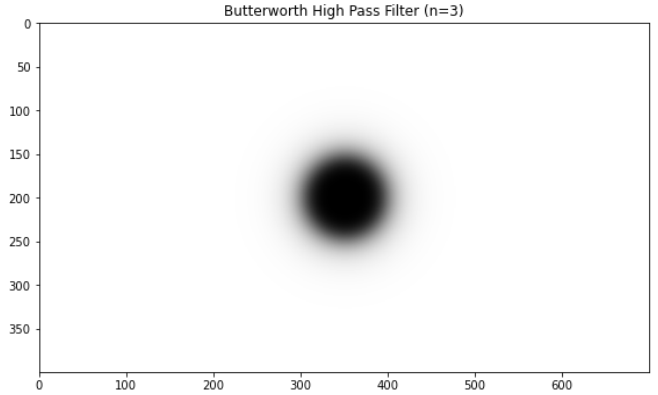


In [9]:
```
plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

LowPass = butterworthLP(50,img.shape,20)
plt.subplot(131), plt.imshow(LowPass, "gray"), plt.title("Butterworth Low Pass Filter (n

HighPass = butterworthHP(50,img.shape,20)
plt.subplot(132), plt.imshow(HighPass, "gray"), plt.title("Butterworth High Pass Filter

plt.show()
```

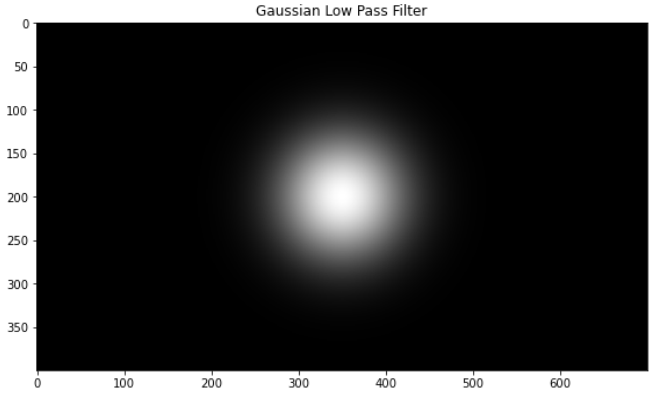


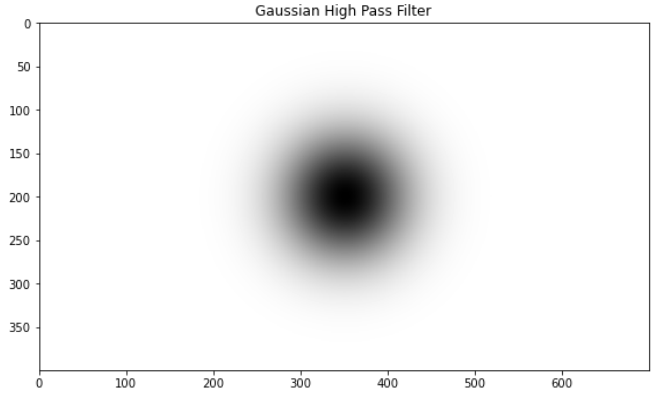
In [10]:
```
plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

LowPass = butterworthLP(50,img.shape,3)
plt.subplot(131), plt.imshow(LowPass, "gray"), plt.title("Butterworth Low Pass Filter (n

HighPass = butterworthHP(50,img.shape,3)
plt.subplot(132), plt.imshow(HighPass, "gray"), plt.title("Butterworth High Pass Filter

plt.show()
```





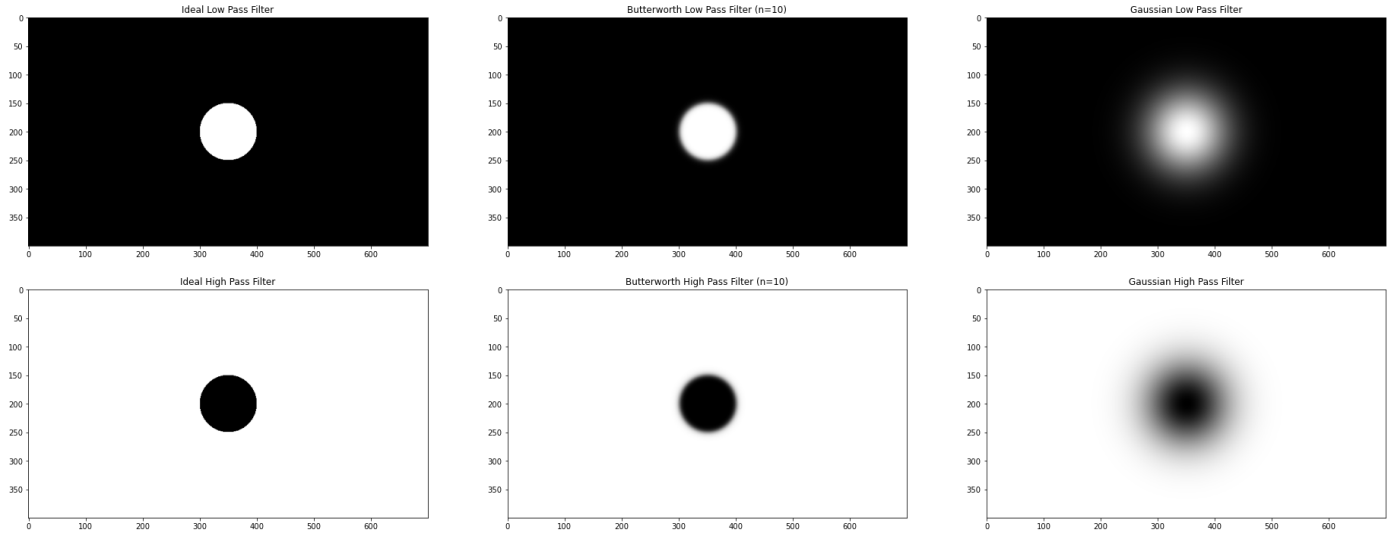
In [11]:
```
plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

LowPass = gaussianLP(50,img.shape)
plt.subplot(131), plt.imshow(LowPass, "gray"), plt.title("Gaussian Low Pass Filter")

HighPass = gaussianHP(50,img.shape)
plt.subplot(132), plt.imshow(HighPass, "gray"), plt.title("Gaussian High Pass Filter")

plt.show()
```



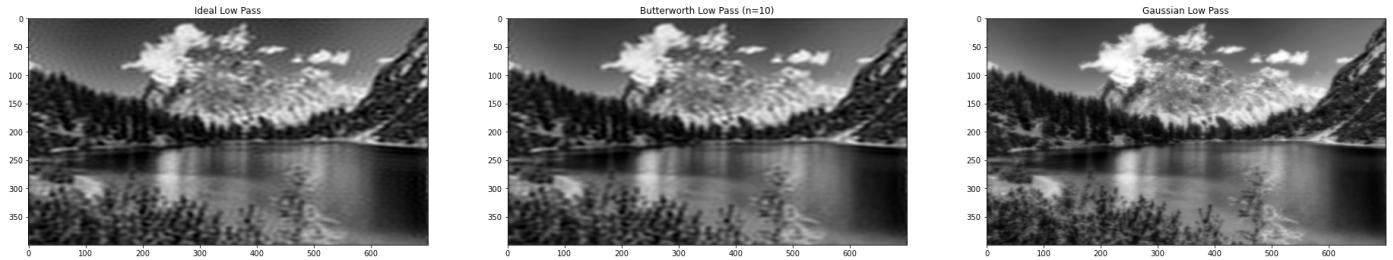


In [ ]:
```
```

In [12]:
```
plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

IdealLP = idealFilterLP(50,img.shape)
plt.subplot(131), plt.imshow(IdealLP, "gray"), plt.title("Ideal Low Pass Filter")

ButterLP = butterworthLP(50,img.shape,10)
plt.subplot(132), plt.imshow(ButterLP, "gray"), plt.title("Butterworth Low Pass Filter (

GaussianLP = gaussianLP(50,img.shape)
plt.subplot(133), plt.imshow(GaussianLP, "gray"), plt.title("Gaussian Low Pass Filter")

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)
IdealHP = idealFilterHP(50,img.shape)
plt.subplot(231), plt.imshow(IdealHP, "gray"), plt.title("Ideal High Pass Filter")

ButterHP = butterworthHP(50,img.shape,10)
plt.subplot(232), plt.imshow(ButterHP, "gray"), plt.title("Butterworth High Pass Filter

GaussianHP = gaussianHP(50,img.shape)
plt.subplot(233), plt.imshow(GaussianHP, "gray"), plt.title("Gaussian High Pass Filter")

plt.show()
```

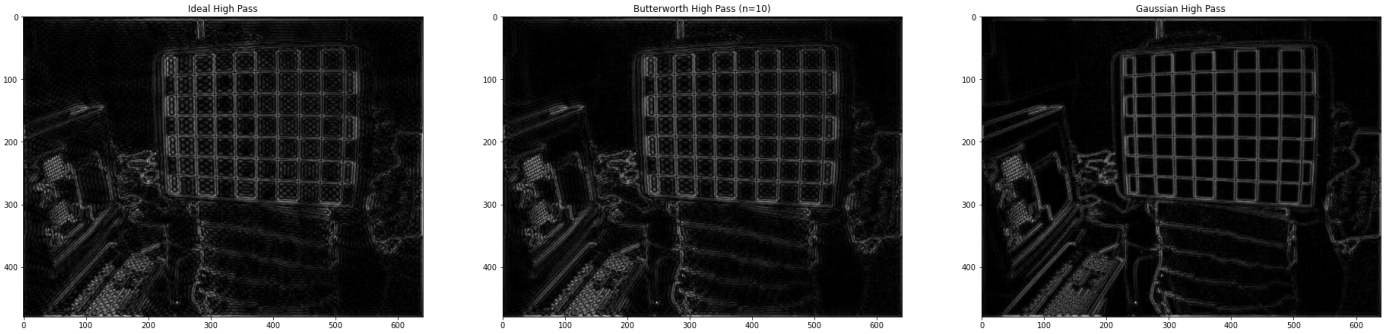


In [13]:
```
img = cv2.imread("data/image.jpg", 0)
original = np.fft.fft2(img)
center = np.fft.fftshift(original)

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

LowPassCenter = center * idealFilterLP(50,img.shape)
LowPass = np.fft.ifftshift(LowPassCenter)
inverse_LowPass = np.fft.ifft2(LowPass)
plt.subplot(131), plt.imshow(np.abs(inverse_LowPass), "gray"), plt.title("Ideal Low Pass

LowPassCenter = center * butterworthLP(50,img.shape,10)
LowPass = np.fft.ifftshift(LowPassCenter)
inverse_LowPass = np.fft.ifft2(LowPass)
plt.subplot(132), plt.imshow(np.abs(inverse_LowPass), "gray"), plt.title("Butterworth Lo

LowPassCenter = center * gaussianLP(50,img.shape)
LowPass = np.fft.ifftshift(LowPassCenter)
inverse_LowPass = np.fft.ifft2(LowPass)
plt.subplot(133), plt.imshow(np.abs(inverse_LowPass), "gray"), plt.title("Gaussian Low P

plt.show()
```



In [14]:
```
img = cv2.imread("data/left01.jpg", 0)
original = np.fft.fft2(img)
center = np.fft.fftshift(original)

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

HighPassCenter = center * idealFilterHP(50,img.shape)
HighPass = np.fft.ifftshift(HighPassCenter)
inverse_HighPass = np.fft.ifft2(HighPass)
plt.subplot(131), plt.imshow(np.abs(inverse_HighPass), "gray"), plt.title("Ideal High Pa

HighPassCenter = center * butterworthHP(50,img.shape,10)
HighPass = np.fft.ifftshift(HighPassCenter)
inverse_HighPass = np.fft.ifft2(HighPass)
plt.subplot(132), plt.imshow(np.abs(inverse_HighPass), "gray"), plt.title("Butterworth H
```

```
HighPassCenter = center * gaussianHP(50,img.shape)
HighPass = np.fft.ifftshift(HighPassCenter)
inverse_HighPass = np.fft.ifft2(HighPass)
plt.subplot(133), plt.imshow(np.abs(inverse_HighPass), "gray"), plt.title("Gaussian High

plt.show()
```



In [15]:

```
img = cv2.imread("data/image.jpg", 0)
original = np.fft.fft2(img)
center = np.fft.fftshift(original)

plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)

plt.subplot(131), plt.imshow(img, "gray"), plt.title("Original Image")

LowPassCenter = center * gaussianLP(50,img.shape)
LowPass = np.fft.ifftshift(LowPassCenter)
inverse_LowPass = np.fft.ifft2(LowPass)
plt.subplot(132), plt.imshow(np.abs(inverse_LowPass), "gray"), plt.title("Gaussian Low P

HighPassCenter = center * gaussianHP(50,img.shape)
HighPass = np.fft.ifftshift(HighPassCenter)
inverse_HighPass = np.fft.ifft2(HighPass)
plt.subplot(133), plt.imshow(np.abs(inverse_HighPass), "gray"), plt.title("Gaussian High

plt.show()
```

Excercises

a. Low Pass Filter application

A data signal has a mixture of different frequency components in it. The frequency contents of the signal and their powers can be obtained through operations such as the Fast Fourier Transform (FFT). A low-pass filter is a filter that passes signals with a frequency lower than a selected cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency. In other words, the frequency components higher than the cutoff frequency will be stopped by a low-pass filter. The exact frequency response of the filter depends on the filter design. Low-pass filters provide a smoother form of a signal, removing the short-term fluctuations and leaving the longer-term trend. Filter designers will often use the low-pass form as a

prototype filter. This type of filter is especially useful since the random errors involved in the raw position data obtained through reconstruction are characterized by relatively high frequency contents. That is, a filter with unity bandwidth and impedance.

Fourier Transform For non-realtime filtering, to achieve a low pass filter, the entire signal is usually taken as a looped signal, the Fourier transform is taken, filtered in the frequency domain, followed by an inverse Fourier transform. Only O(n log(n)) operations are required compared to O(n2) for the time domain filtering algorithm. This can also sometimes be done in real-time, where the signal is delayed long enough to perform the Fourier transformation on shorter, overlapping blocks.

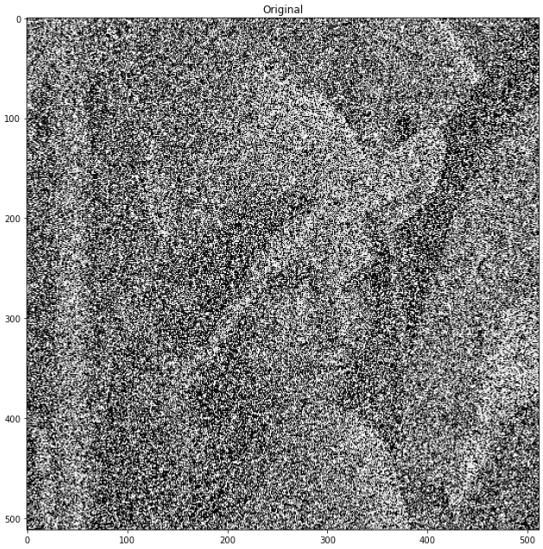
In [16]:
```
##########################################
# Low Pass Filter Applications
##########################################

def lowPassFilters(img, D0=20, n=10):
    """
    Takes an image, and a distance D0 as well as an n for the exponentaial parameter of
    Returns a collection of images:
    Original Image / Fourier Spectrum
    Ideal Filter / Buttlesworth Filter / Gaussian Filter
    """

    def distance(point1, point2):
        """
        Returns the euclidean distance between two points.
        """
        return sqrt((point1[0]-point2[0])**2 + (point1[1]-point2[1])**2)

    def idealFilter(D0, imgShape):
        """
        It generates a zero matrix and change the values on a nested loop of the distanc
        Rule is: preservation of pixels that are on a D0 distance from the center throug
        Tends to create ringins due to the abrupt attenuation of high frequency parts.
        """
        base = np.zeros(imgShape[:2])
        rows, cols = imgShape[:2]
        center = (rows/2, cols/2)
        for x in range(cols):
            for y in range(rows):
                if distance((y, x), center) < D0:
                    base[y, x] = 1
        return base

    def butterworth(D0, imgShape, n):
        """
        It generates a zero matrix and change the values on a nested loop of the distanc
        Rule is: Reduces the intensity of pixels absed on their distance to D0, if dista
        As the distance expands the ratio 1/([1+D(u,v)/D0]2**n) tends to crete lower val
        with a smooth transition that relies heavily on the exponent n.
        """
        base = np.zeros(imgShape[:2])
        rows, cols = imgShape[:2]
        center = (rows/2, cols/2)
        for x in range(cols):
            for y in range(rows):
                base[y, x] = 1/(1+(distance((y, x), center)/D0)**(2*n))
        return base

    def gaussian(D0, imgShape):
        """
        It generates a zero matrix and change the values on a nested loop of the distanc
        Rule is: Quite simmilar to Buttlesworth but it uses constant e by the power of a
```

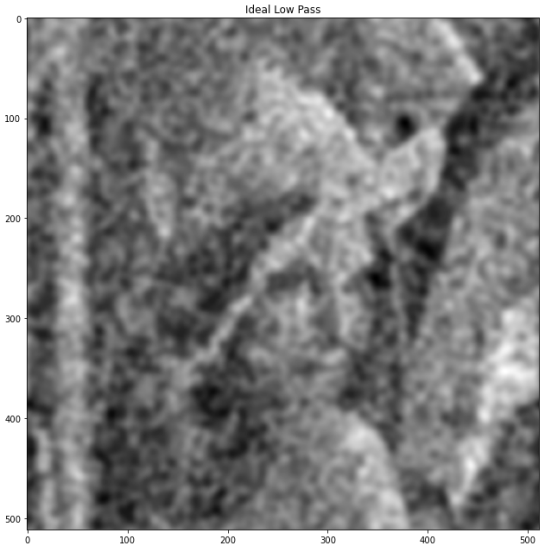
```
        Where e**0 == 1 and from them but this tends to decay by the exponent -(D**2(u,v
        This also makes a smoother transition between low and attunned high frequencies.
        """
        base = np.zeros(imgShape[:2])
        rows, cols = imgShape[:2]
        center = (rows/2, cols/2)
        for x in range(cols):
            for y in range(rows):
                base[y, x] = exp(((-distance((y, x), center)**2)/(2*(D0**2))))
        return base

    # Spectrum of provided image
    spectrum = np.fft.fftshift(np.fft.fft2(img))

    # Ideal filter mask
    LowPassCenter = spectrum * idealFilter(D0, img.shape)
    # Back from frequency
    LowPass = np.fft.ifftshift(LowPassCenter)
    inverse_LowPass = np.fft.ifft2(LowPass)

    # ButterWorth's filter mask
    LowPassCenter = spectrum * butterworth(D0, img.shape, n)
    # Back from frequency
    LowPass = np.fft.ifftshift(LowPassCenter)
    inverse_LowPass = np.fft.ifft2(LowPass)

    # Gaussian's filter mask
    LowPassCenter = spectrum * gaussian(D0, img.shape)
    # Back from Frequency
    LowPass = np.fft.ifftshift(LowPassCenter)
    inverse_LowPass = np.fft.ifft2(LowPass)

    # Plot images
    fig = plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)
    plt.subplot(221), plt.imshow(np.abs(img), "gray"), plt.title("Original")
    plt.subplot(222), plt.imshow(np.abs(inverse_LowPass),"gray"), plt.title("Ideal Low P
    plt.subplot(223), plt.imshow(np.abs(inverse_LowPass),"gray"), plt.title(f"Butterwort
    plt.subplot(224), plt.imshow(np.abs(inverse_LowPass),"gray"), plt.title("Gaussian Lo
    fig.suptitle(f"Low Pass FTT Filtering, using a distance to center of {D0} and an n o
    plt.show()
```

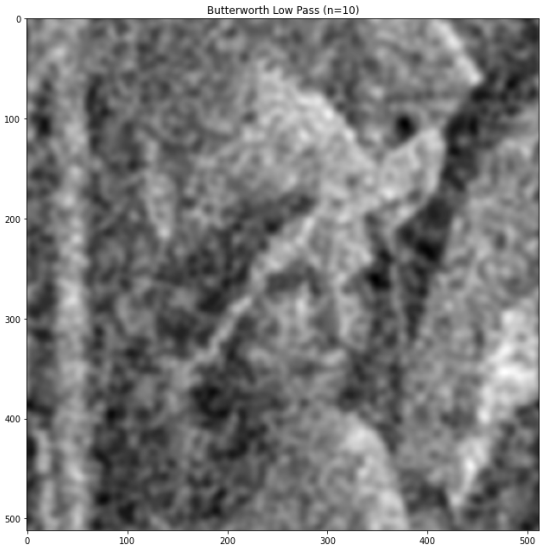
Here we have how lopass is good to reduce noise

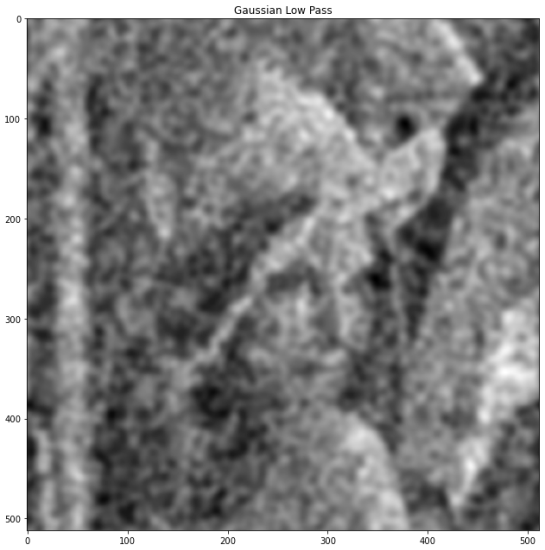
In [17]:
```
# Denoise application
img = cv2.imread("data/lena_supernoisy.jpg", 0)
lowPassFilters(img, D0=20, n=10)
```

Low Pass FTT Filtering, using a distance to center of 20 and an n of 10





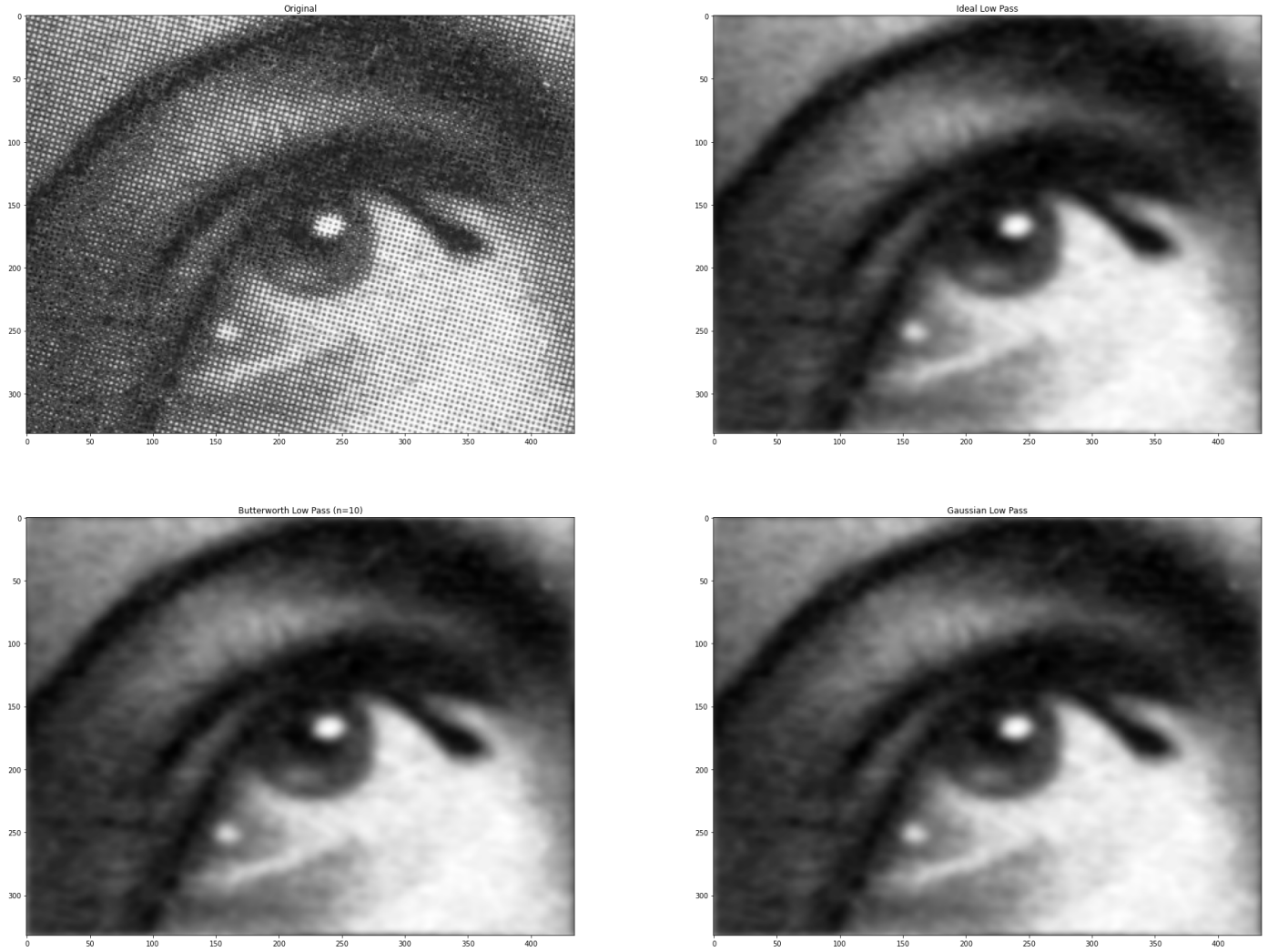




Another application examples to smooth images

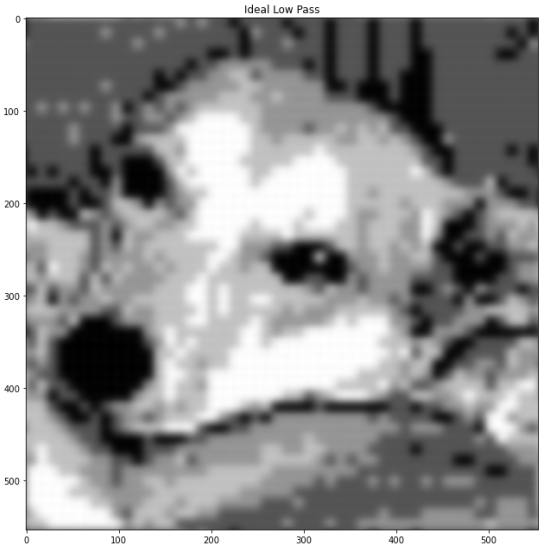
In [18]:
```
# Smooth application
img = cv2.imread("data/eye.jpg", 0)
lowPassFilters(img, D0=20, n=10)
```

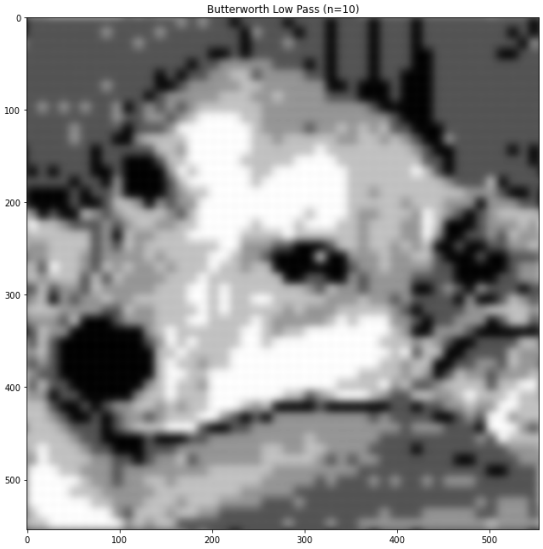
Low Pass FTT Filtering, using a distance to center of 20 and an n of 10

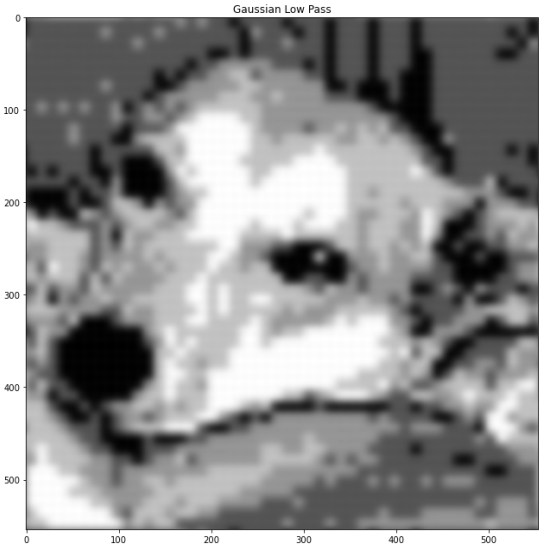


In [19]:
```
# Smooth application
img = cv2.imread("data/lego_perro.jpg", 0)
lowPassFilters(img, D0=20, n=10)
```

Low Pass FTT Filtering, using a distance to center of 20 and an n of 10







An exaplanation of each of the filters:

Ideal Filter

An ideal filter is a frequency selective network that has very sharp cut-off characteristics, it transmits the signals of certain specified band of frequencies exactly and totally rejects the signals of frequencies outside this band.

An ideal low pass filter is the one which transmits all the signal of frequencies less than a cut-off frequency (or radians per second) without any distortion and blocks all the signals of frequencies above it.

Gaussian Filter

A Gaussian Filter is a low pass filter used for reducing noise (high frequency components) and blurring regions of an image. The filter is implemented as an Odd sized Symmetric Kernel which is passed through each pixel of the Region of Interest to get the desired effect.

The kernel is not hard towards drastic color changed (or edges) due to it the pixels towards the center of the kernel having more weightage towards the final value then the periphery.

Butterworth Filter

The Butterworth filter is an analogue filter design which produces the best output response with no ripple in the pass band or the stop band resulting in a maximally flat filter response but at the expense of a relatively wide transition band.

The frequency response of the Butterworth Filter approximation function is also often referred to as “maximally flat” (no ripples) response because the pass band is designed to have a frequency response which is as flat as mathematically possible.

However, one main disadvantage of the Butterworth filter is that it achieves this pass band flatness at the expense of a wide transition band as the filter changes from the pass band to the stop band. It also has poor phase characteristics as well.

b. High Pass Filter application

Some applications of a high-pass filter in the Fourier transform of an image are:

- Edge detection: The high-pass filters can be used to enchance the edges inside an image, making them easier to detect and to analyze them in a better way regards the function.
- Image denoising: This is a common technique in image processing, separate the high-frequency components inside an image, which contain the detail and texture information, from its low-frequency components, which contain the smooth background information and the noise.
- Image sharpening: As the name says it, can be used to sharpen the details in an image, making it appear clearer and more defined.
- Image segmentation: Which is to separate objects or regions of interest in an image. Basiclly is to use the high-frequency components of the image to highlight boundaries between objects, and then use this information to segment the image as needed.
- Medical imaging: This is used to enhance certain features or to reduce noise to have a better image to analize. Some high-pass filters used in this area are Compute Tomography scans, Ultrasounf imaging and X-Ray Imaging.
- Noise reduction: Used to remove unwanted noise from an image, suppress low-frequency noise inside an image making the image clearer and smoother.
- Feature extraction: It can be used as well to extract high-frequency features,like edges and corners for further analysis and processing.

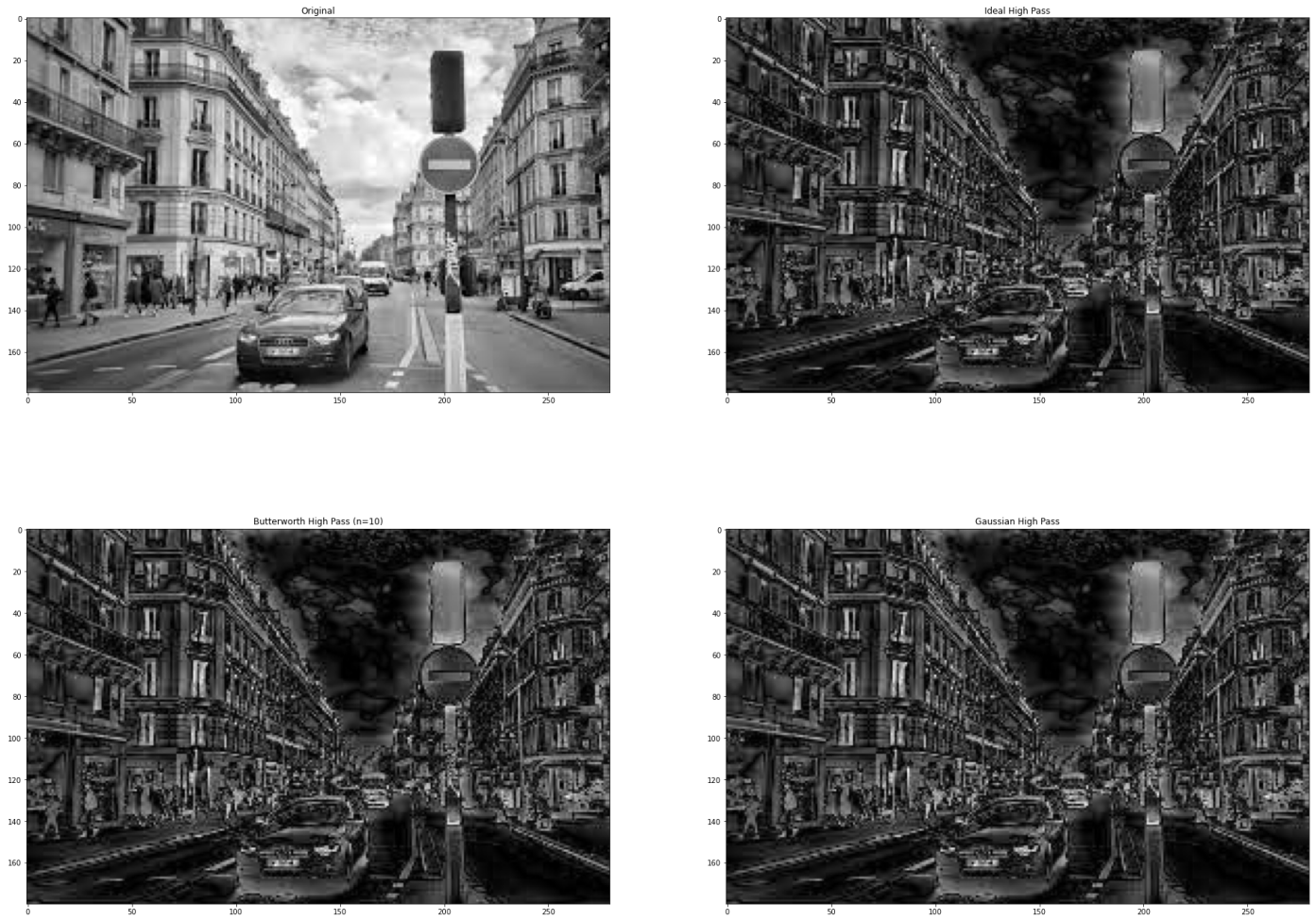
In [20]:
```
##########################################
# High Pass Filter Applications
##########################################

def highPassFilters(img, D0=5, n=10, img_pos=[221, 222, 223, 224]):
    """
    Takes an image, and a distance D0 as well as an n for the exponentaial parameter of
    Returns a collection of images:
    Original Image / Fourier Spectrum
    Ideal Filter / Buttlesworth Filter / Gaussian Filter
    """

    def idealFilterHP(D0,imgShape):
```

```
        base = np.ones(imgShape[:2])
        rows, cols = imgShape[:2]
        center = (rows/2,cols/2)
        for x in range(cols):
            for y in range(rows):
                if distance((y,x),center) < D0:
                    base[y,x] = 0
        return base

    def butterworthHP(D0,imgShape,n):
        base = np.zeros(imgShape[:2])
        rows, cols = imgShape[:2]
        center = (rows/2,cols/2)
        for x in range(cols):
            for y in range(rows):
                base[y,x] = 1-1/(1+(distance((y,x),center)/D0)**(2*n))
        return base

    def gaussianHP(D0,imgShape):
        base = np.zeros(imgShape[:2])
        rows, cols = imgShape[:2]
        center = (rows/2,cols/2)
        for x in range(cols):
            for y in range(rows):
                base[y,x] = 1 - exp(((-distance((y,x),center)**2)/(2*(D0**2))))
        return base

    original = np.fft.fft2(img)
    center = np.fft.fftshift(original)

    # Ideal Filter
    HighPassCenter = center * idealFilterHP(D0,img.shape)
    HighPass = np.fft.ifftshift(HighPassCenter)
    inverse_HighPass = np.fft.ifft2(HighPass)

    # Butterworth Filter
    HighPassCenter = center * butterworthHP(D0,img.shape,n)
    HighPass = np.fft.ifftshift(HighPassCenter)
    inverse_HighPass = np.fft.ifft2(HighPass)

    # Gaussian Filter
    HighPassCenter = center * gaussianHP(D0,img.shape)
    HighPass = np.fft.ifftshift(HighPassCenter)
    inverse_HighPass = np.fft.ifft2(HighPass)

    fig = plt.figure(figsize=(6.4*5, 4.8*5), constrained_layout=False)
    plt.subplot(img_pos[0]), plt.imshow(np.abs(img), "gray"), plt.title("Original")
    plt.subplot(img_pos[1]), plt.imshow(np.abs(inverse_HighPass), "gray"), plt.title("Id
    plt.subplot(img_pos[2]), plt.imshow(np.abs(inverse_HighPass), "gray"), plt.title(f"B
    plt.subplot(img_pos[3]), plt.imshow(np.abs(inverse_HighPass), "gray"), plt.title("Ga
    fig.suptitle(f"High Pass FTT Filtering, using a distance to center of {D0} and an n
    plt.show()
```

In [26]:

```
# Edge detection for ADAS applications
img = cv2.imread("data/road.jpeg", 0)
highPassFilters(img, 5, 10)
```

High Pass FTT Filtering, using a distance to center of 5 and an n of 10



In our example we can see how through using a High Pass filter in the Fourier transform over an image taken from the road used for various Advanced Driver Assistance Systems (ADAS) applications. One of its most common applications is to sharpen the edges and details of the image, making it useful in computer vision tasks such as object detection and recognition.

High-pass filters in the Fourier domain function by removing the low-frequency components of an image and keeping the high-frequency components. The low-frequency components represent the smooth, uniform areas of the image while the high-frequency components represent the edges and details. By eliminating the low-frequency components and boosting the high-frequency components, a high-pass filter can improve the visibility of the edges and details, enhancing the performance of computer vision algorithms that rely on these features [1].

In [22]:
```
# Medical imaging application
img = cv2.imread("data/hand_xray.jpg", 0)
highPassFilters(img, 2, 10)
```

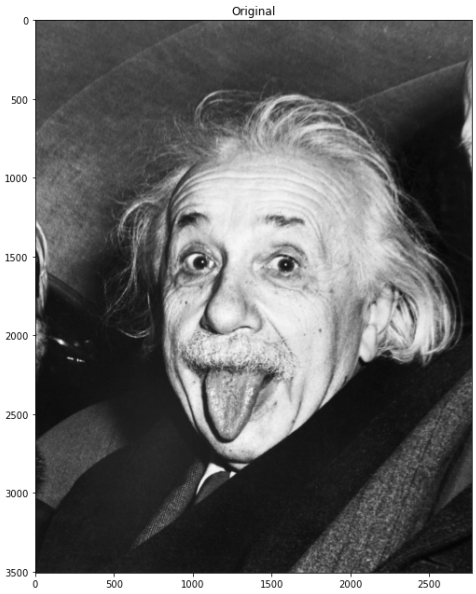
High Pass FTT Filtering, using a distance to center of 2 and an n of 10

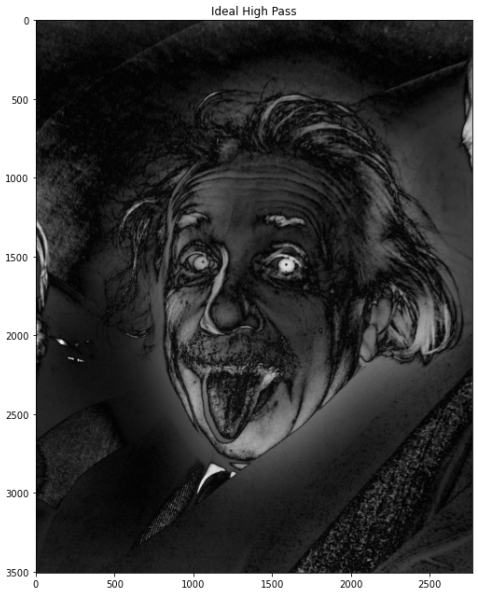
In our example we can see how through using a High Pass filter in the Fourier transform over an x-Ray image.We can see the use of the filter enhance the visibility and reduces a little bit of the noise that the X-Ray had in the begining. By removing the low-frequency noise, the high-pass filter can help to better highlight the important features of the X-Ray if needed.

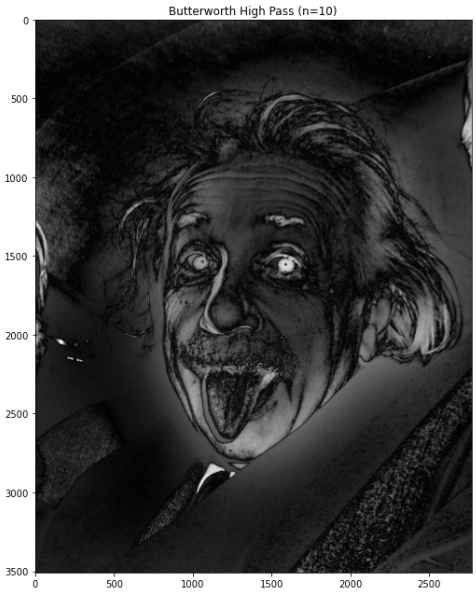
However, it is important to carefully choose the cutoff frequency for the high-pass filter to avoid removing important low-frequency information from the image. With a different example that may need more definition with a smaller section, this could be of great help to analyze and verify data for the user, but this technique may not always be suitable for all types of X-Ray images depending on the use.

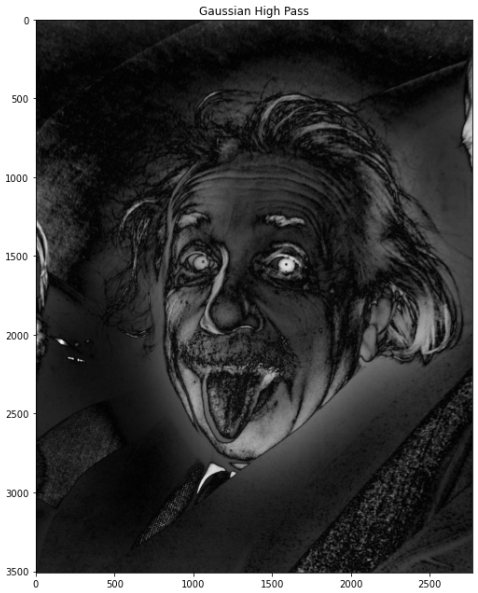
In [23]:
```
# For image sengmentation and fun
img = cv2.imread("data/albert.jpeg", 0)
highPassFilters(img, 2, 10)
```

High Pass FTT Filtering, using a distance to center of 2 and an n of 10









References

[1] (2020, December 26). Apply a Gauss filter to an image with python. GeeksforGeeks. Retrieved February 11, 2023, from https://www.geeksforgeeks.org/apply-a-gauss-filter-to-an-image-with-python/

[2] Bevelacqua, P. (n.d.). Signal Processing: Filtering. TheFourierT

[3] Characteristics of an ideal filter (LPF HPF BPF and BRF). TutorialsPoint. (n.d.). Retrieved February 11, 2023, from https://www.tutorialspoint.com/characteristics-of-an-ideal-filter-lpf-hpf-bpf-and-brf#:~:text=An%20ideal%20low%20pass%20filter,%CF%89c%20radians%20per%20second.

[4] Dogra A, Bhalla P. Image Sharpening By Gaussian And Butterworth High Pass Filter. Biomed Pharmacol J 2014;7(2). Available from: http://biomedpharmajournal.org/?p=3274 ransform.com - Signal Processing: Filtering. Retrieved February 12, 2023, from https://www.thefouriertransform.com/applications/filtering.php

[5] Gonzalez, R. C., & Woods, R. E. (2008). Digital Image Processing (3rd ed.). Pearson Education.

[6] Low-pass filter. Definition of Low-Pass Filter | Analog Devices. (n.d.). Retrieved February 11, 2023, from https://www.analog.com/en/design-center/glossary/low-pass-filter.html#:~:text=A%20low%2Dpass%20filter%20(LPF,attenuates%20all%20signals%20below%20it.

[7] Rosebrock, A. (2022, February 6). OpenCV fast fourier transform (FFT) for Blur detection in images and video streams. PyImageSearch. Retrieved February 11, 2023, from https://pyimagesearch.com/2020/06/15/opencv-fast-fourier-transform-fft-for-blur-detection-in-images-and-video-streams/

[8] Storr, W. (2022, August 6). Butterworth filter design and low Pass Butterworth filters. Basic Electronics Tutorials. Retrieved February 11, 2023, from https://www.electronics-tutorials.ws/filter/filter_8.html