Fourier Transform

In [7]:
```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

In [3]:
```
def apply_fourier_transform(image):
    f_transform = np.fft.fft2(image)
    f_shift = np.fft.fftshift(f_transform)
    magnitude_spectrum = 20 * np.log(np.abs(f_shift) + 1)  # +1 to avoid log(0)
    return magnitude_spectrum, f_shift
```

Butterworth Filter

In [4]:
```
def butterworth_filter(shape, cutoff, order):
    P, Q = shape
    u = np.arange(P) - P // 2
    v = np.arange(Q) - Q // 2
    U, V = np.meshgrid(u, v)
    D = np.sqrt(U**2 + V**2)
    H = 1 / (1 + (D / cutoff)**(2 * order))
    return H
```

Gaussian Filter

In [5]:
```
def gaussian_filter(shape, cutoff):
    P, Q = shape
    u = np.arange(P) - P // 2
    v = np.arange(Q) - Q // 2
    U, V = np.meshgrid(u, v)
    D = np.sqrt(U**2 + V**2)
    H = np.exp(-(D**2) / (2 * (cutoff**2)))
    return H

# Apply filters in the Fourier domain
def apply_filter_in_fourier(image, filter):
    _, f_shift = apply_fourier_transform(image)

    # Ensure the filter shape matches the image shape
    if filter.shape != f_shift.shape:
        filter = filter.T  # Transpose filter to match the shape if needed

    filtered_shift = f_shift * filter
    f_ishift = np.fft.ifftshift(filtered_shift)
    filtered_image = np.abs(np.fft.ifft2(f_ishift))
    return filtered_image
```

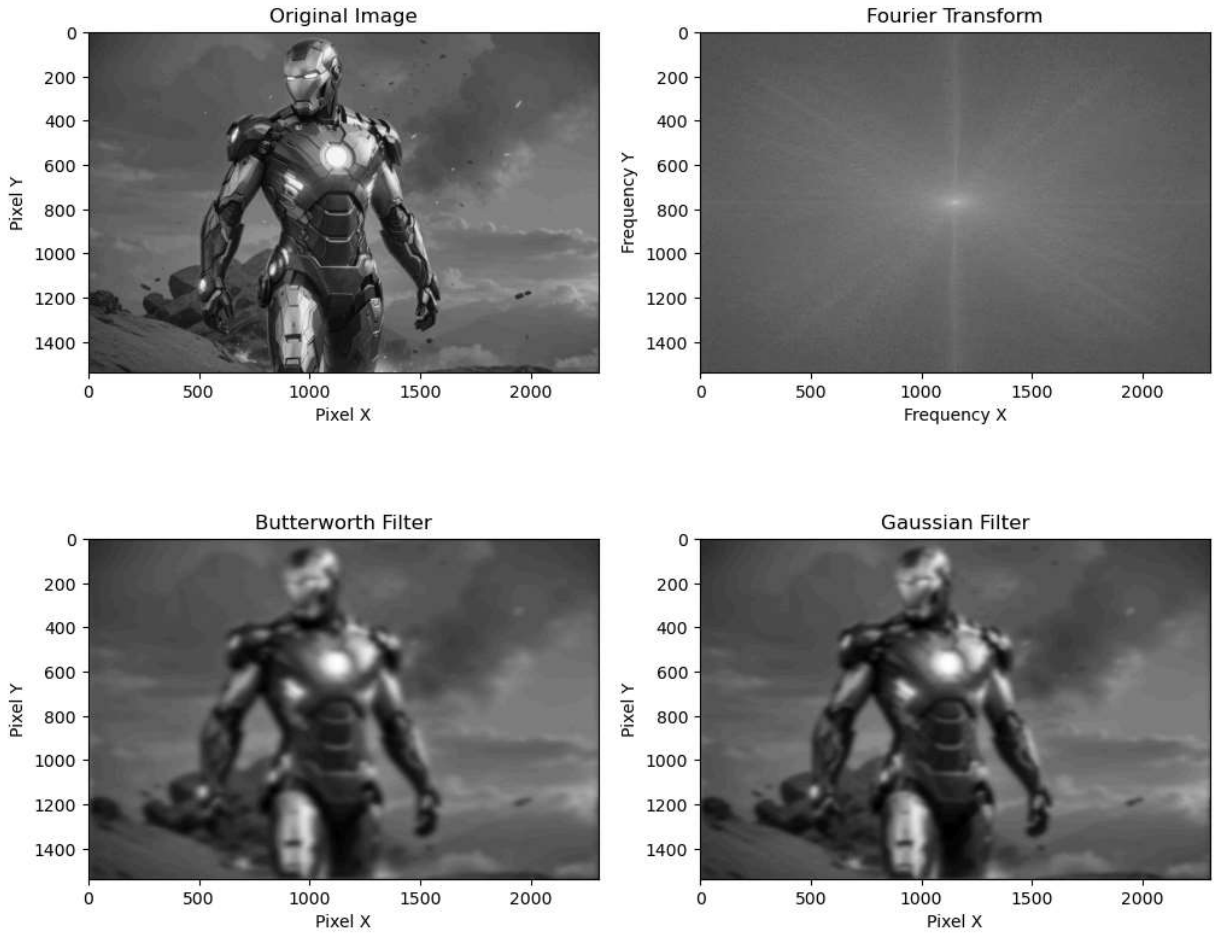
In [8]:
```
# Load and convert the image to grayscale
image = cv2.imread('img.jpg', cv2.IMREAD_GRAYSCALE)
```

In [9]:
```
# Apply Fourier Transform
magnitude_spectrum, _ = apply_fourier_transform(image)
```

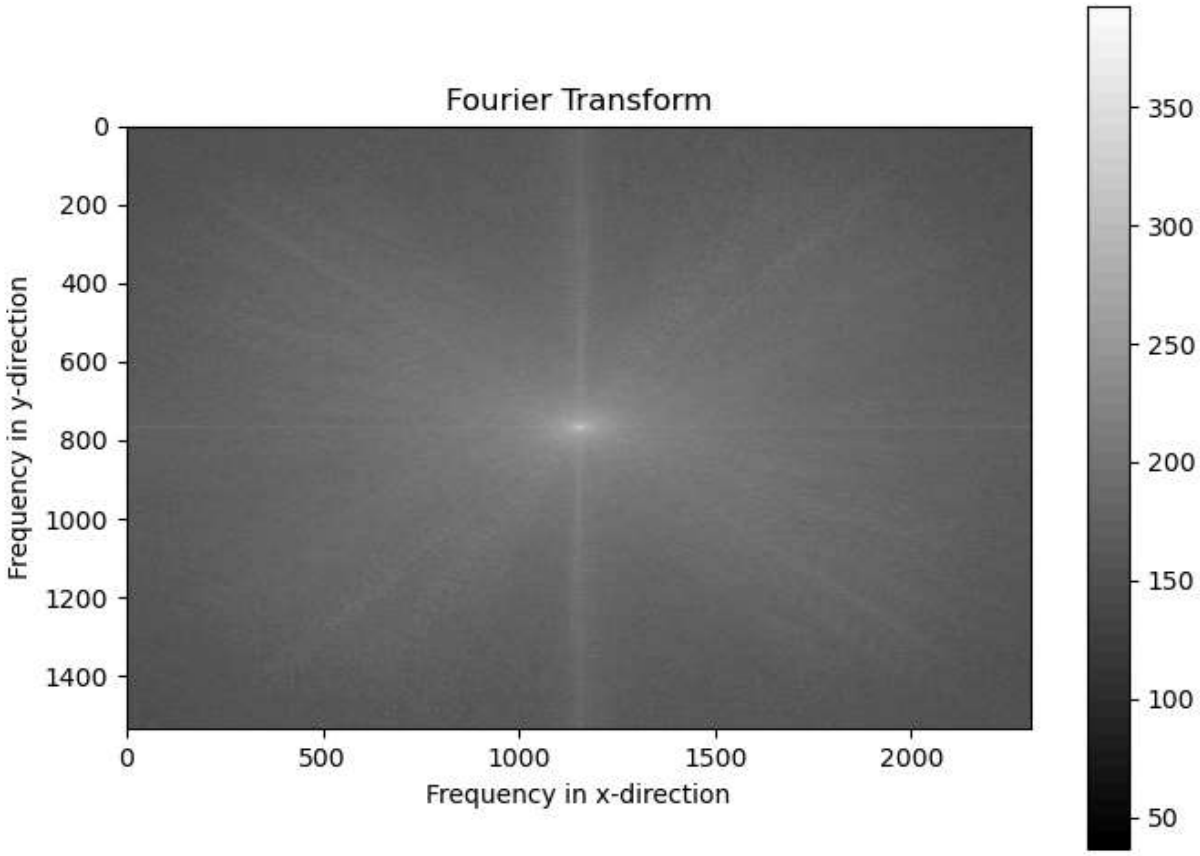
```
# Apply Butterworth and Gaussian filters
butterworth = butterworth_filter(image.shape, cutoff=30, order=2)
gaussian = gaussian_filter(image.shape, cutoff=30)

butterworth_result = apply_filter_in_fourier(image, butterworth)
gaussian_result = apply_filter_in_fourier(image, gaussian)
```

In [10]:
```
plt.figure(figsize=(12, 10))

# Display Original Image
plt.subplot(2, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.xlabel('Pixel X')
plt.ylabel('Pixel Y')

# Display Fourier Transform
plt.subplot(2, 2, 2)
plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('Fourier Transform')
plt.xlabel('Frequency X')
plt.ylabel('Frequency Y')

# Display Butterworth Filtered Image
plt.subplot(2, 2, 3)
plt.imshow(butterworth_result, cmap='gray')
plt.title('Butterworth Filter')
plt.xlabel('Pixel X')
plt.ylabel('Pixel Y')

# Display Gaussian Filtered Image
plt.subplot(2, 2, 4)
plt.imshow(gaussian_result, cmap='gray')
plt.title('Gaussian Filter')
plt.xlabel('Pixel X')
plt.ylabel('Pixel Y')
```

Out[10]:
```
Text(0, 0.5, 'Pixel Y')
```



In [11]:
```
plt.figure(figsize=(8, 6))
plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('Fourier Transform')

# Adding x and y axis labels
plt.xlabel('Frequency in x-direction')
plt.ylabel('Frequency in y-direction')

plt.colorbar()  # Show color bar for the magnitude values
plt.show()
```


Fourier Transform
Frequency in y-direction
Frequency in x-direction
0
200
400
600
800
1000
1200
1400
0
500
1000
1500
2000
350
300
250
200
150
100
50

In [ ]: