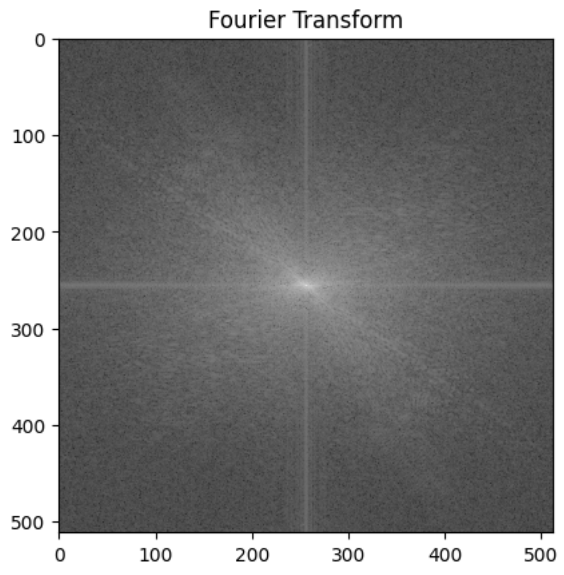
4. Take any image and apply the Fourier Transform to this image and the following filters:(Python or MATLAB) (b) Butterworth filters (c) Gaussian filters

```
import numpy as np
import cv2
from matplotlib import pyplot as plt

# Load image
img = cv2.imread('/content/lena.png', 0)

# Apply Fourier Transform
f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
magnitude_spectrum = 20 * np.log(np.abs(fshift))

plt.imshow(magnitude_spectrum, cmap='gray')
plt.title('Fourier Transform')
plt.show()
```



Fourier Transform of the Image:

The Fourier Transform breaks an image into its frequency components, showing how image details vary across space.

When you apply np.fft.fft2(img), it creates a complex array representing the image's frequency spectrum. fftshift(f) moves the zero frequency (DC) to the center for easier viewing.

The Magnitude Spectrum is calculated using 20 * np.log(np.abs(fshift)), which shows the strength of the frequency components on a log scale for better visualization.

Output:

The magnitude spectrum shows high frequencies near the edges and low frequencies in the center. Bright spots represent strong frequencies, which are linked to repeating patterns or sharp edges in the image.

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

butterworth = butterworth_filter(img.shape, 30, 2)
filtered_image = np.fft.ifftshift(fshift * butterworth)
filtered_image = np.fft.ifft2(filtered_image)
```

Butterworth Filter:

A Butterworth filter is a frequency filter that smoothly transitions from keeping low frequencies (passband) to reducing high frequencies (stopband). The butterworth_filter function creates a low-pass filter where frequencies higher than the cutoff are gradually reduced, with the steepness controlled by the order value.

Effect on the Image:

Smoothing: Applying the Butterworth filter keeps the low-frequency parts of the image (smooth areas) and reduces the high-frequency parts (like sharp edges and noise).

Blurring: The output image will look blurred because the Butterworth filter reduces the high-frequency details, which are responsible for sharpness.

Order and Cutoff: A higher filter order makes the transition from low to high frequencies sharper, making the filter more effective at removing high frequencies. A lower cutoff value removes more high-frequency details, making the image blurrier.

Output:

The filtered image will be a smoother, blurred version of the original. High-frequency details like edges will be softened, but the main structure of the image will remain. The amount of smoothing depends on the cutoff frequency and filter order.

```
from scipy.ndimage import gaussian_filter

# Apply Gaussian filter in Fourier domain
gaussian_filtered = gaussian_filter(img, sigma=5)
```

Gaussian Filter (in Fourier Domain):

A Gaussian filter smooths an image by averaging pixel values based on a Gaussian distribution. It's applied directly in the spatial domain using scipy.ndimage.gaussian_filter.

Effect on the Image:

Smoothing: A Gaussian filter smooths the image by blurring sharp edges and reducing noise. The sigma value controls how much blurring happens, larger sigma values cause more blurring.

```
Start coding or generate with AI.
```