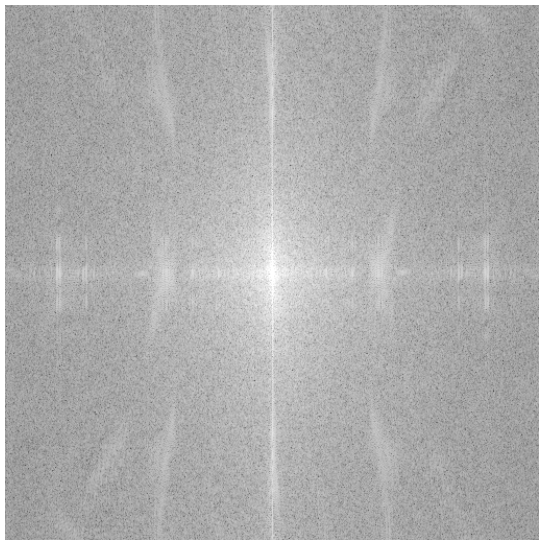
```python
import sys
import cv2
import numpy as np
from numpy import fft


def run():
    filename = sys.argv[-1]
    img = cv2.imread(filename, 0)
    name = filename.split('.')[0]

    in_magnitude_img = get_magnitude(img)
    cv2.imwrite(name + '_magnitudue.png', in_magnitude_img)

    lp_filtered_img = apply_low_pass_filter(img)
    lp_magnitude_img = get_magnitude(lp_filtered_img)
    cv2.imwrite(name + '_filtered_LP.png', lp_filtered_img)
    cv2.imwrite(name + '_magnitude_LP.png', lp_magnitude_img)

    hp_filtered_img = apply_high_pass_filter(img)
    hp_magnitude_img = get_magnitude(hp_filtered_img)
    cv2.imwrite(name + '_filtered_HP.png', hp_filtered_img)
    cv2.imwrite(name + '_magnitude_HP.png', hp_magnitude_img)


def get_magnitude(img):
    fourier = fft.fft2(img)
    shifted_fouirer = fft.fftshift(fourier)
    return 20 * np.log(np.abs(shifted_fouirer))


def apply_low_pass_filter(img):
    lp_filter = generate_lp_filter(img, 50)

    fourier = fft.fft2(img)
    shifted_fouirer = fft.fftshift(fourier)
    phase_img = np.angle(shifted_fouirer)

    lp_filtered_fourier = np.multiply(np.abs(shifted_fouirer), lp_filter)
    recon_img_lp = np.multiply(lp_filtered_fourier, np.exp(1j * phase_img))
    return np.minimum(np.abs(np.real(fft.ifft2(fft.fftshift(recon_img_lp)))), 255)


def generate_lp_filter(img, kernel_size):
    height, width = img.shape
    kernel = np.ones((kernel_size, kernel_size))
    return np.pad(kernel,
                  ((height//2 - kernel_size//2, width//2 - kernel_size//2),
                   (height//2 - kernel_size//2, width//2 - kernel_size//2)),
                  'constant')


def apply_high_pass_filter(img):
    height, width = img.shape
    hp_filter = np.ones((height, width)) - generate_lp_filter(img, 50)

    fourier = fft.fft2(img)
    shifted_fouirer = fft.fftshift(fourier)
    phase_img = np.angle(shifted_fouirer)

    hp_filtered_fourier = np.multiply(np.abs(shifted_fouirer), hp_filter)
    recon_img_hp = np.multiply(hp_filtered_fourier, np.exp(1j * phase_img))
    return np.minimum(np.abs(np.real(fft.ifft2(fft.fftshift(recon_img_hp)))), 255)


run()
```

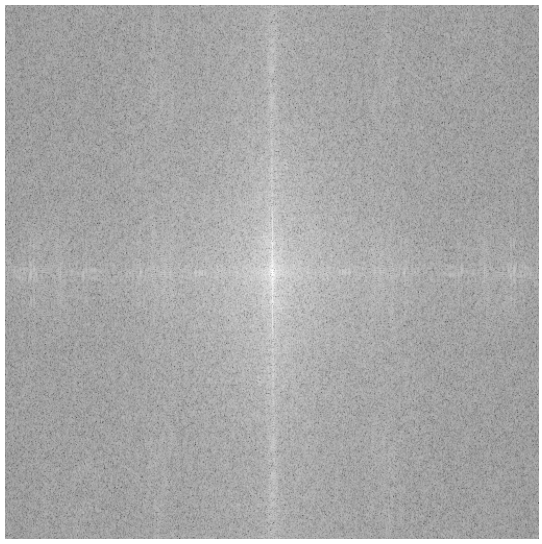
Origin

Origin Magnitude



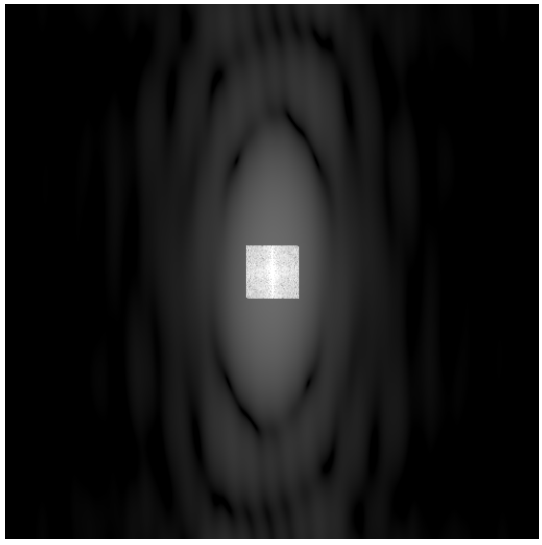
HP Filtered

HP Magnitude



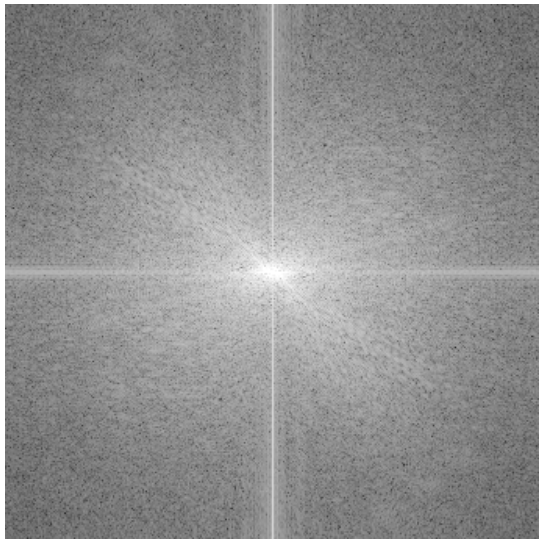
LP Filtered

LP Magnitude

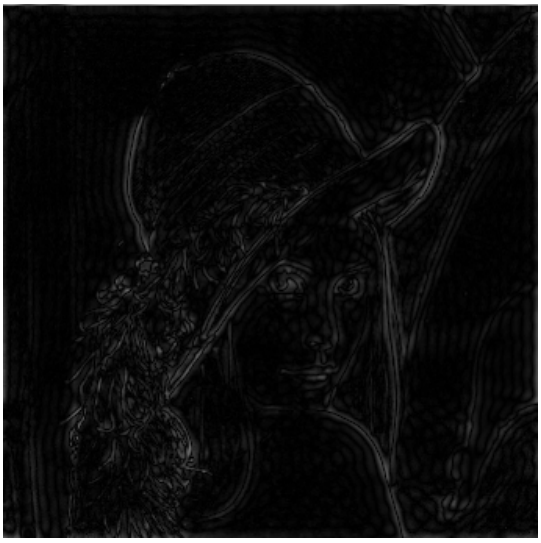


Origin

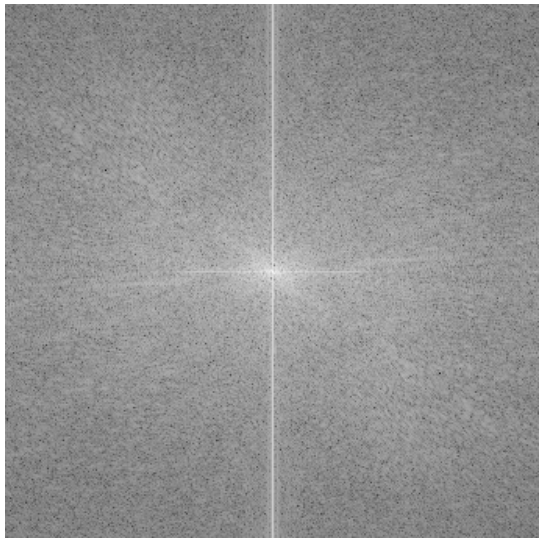
Origin Magnitude



HP Filtered



HP Magnitude



LP Filtered

LP Magnitude