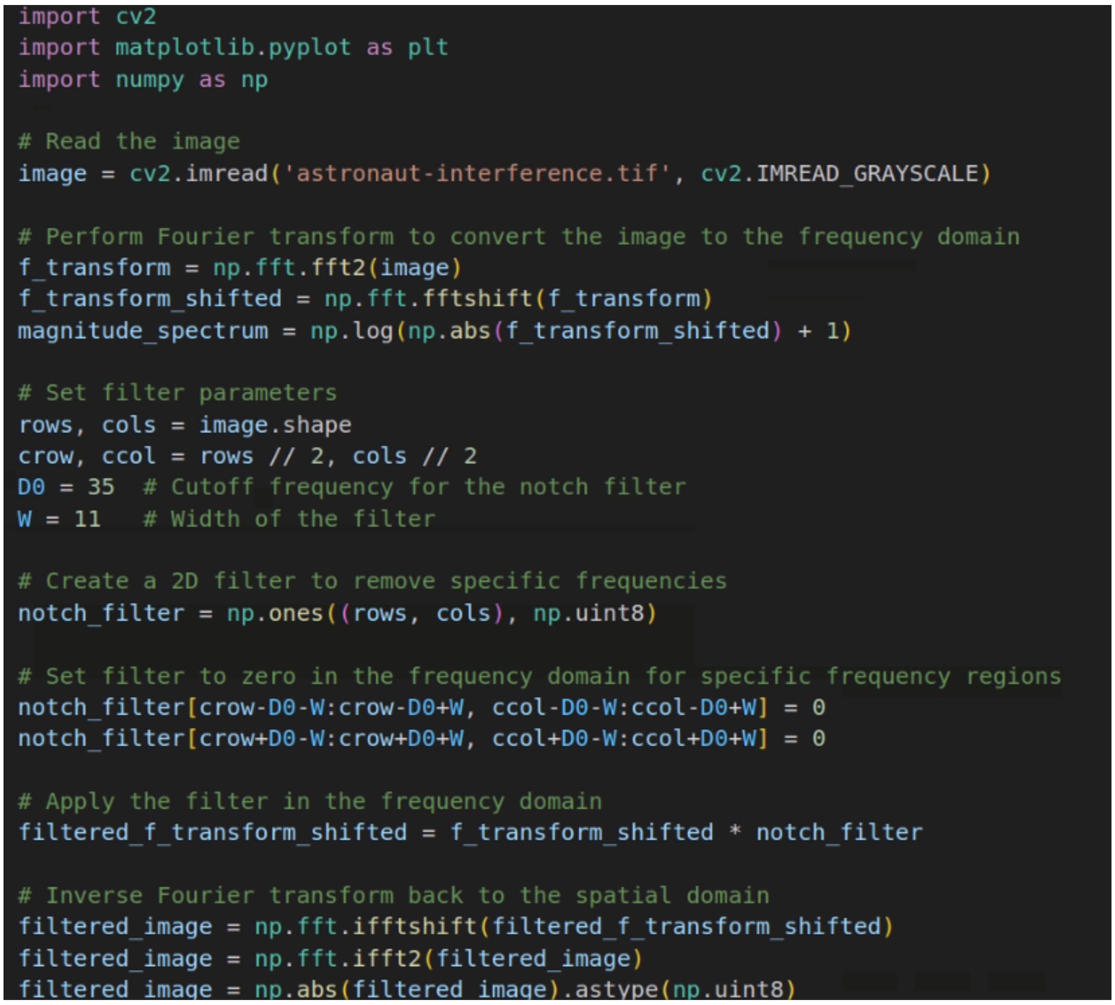
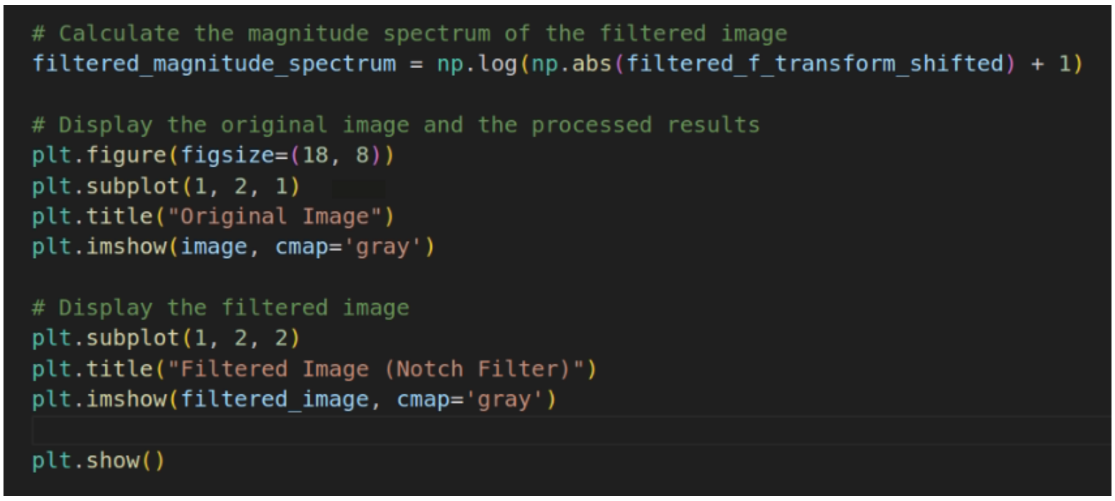
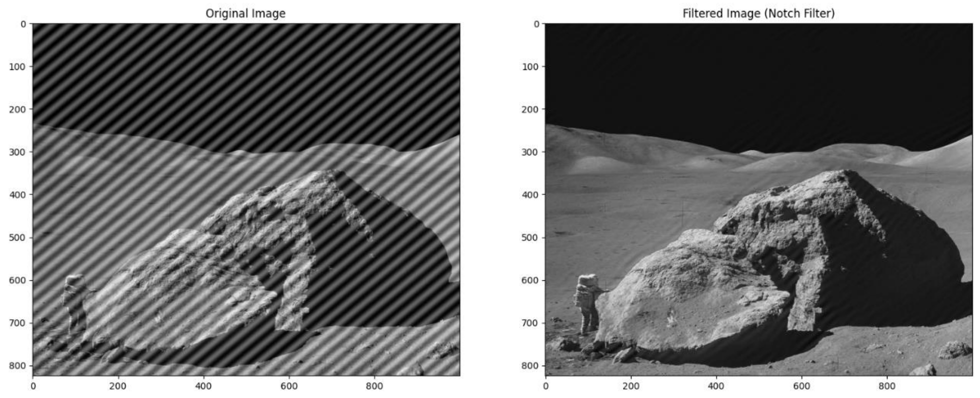
1. Please use FFT and design a frequency filter to cancel the sinusoidal noise of the assigned image, ‘astronaut-interference.tif’, and print out the source code and the processed image? (40)



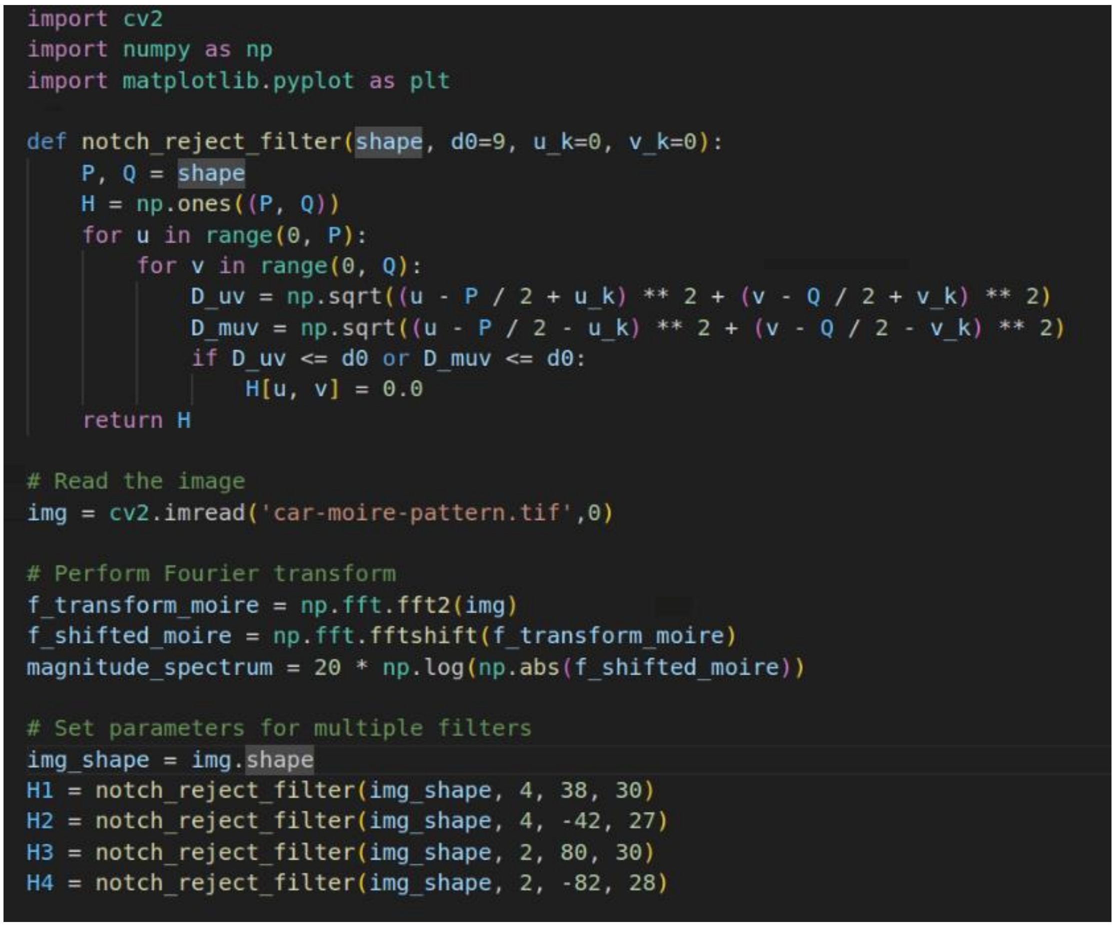
```python
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Read the image
image = cv2.imread('astronaut-interference.tif', cv2.IMREAD_GRAYSCALE)

# Perform Fourier transform to convert the image to the frequency domain
f_transform = np.fft.fft2(image)
f_transform_shifted = np.fft.fftshift(f_transform)
magnitude_spectrum = np.log(np.abs(f_transform_shifted) + 1)

# Set filter parameters
rows, cols = image.shape
crow, ccol = rows // 2, cols // 2
D0 = 35  # Cutoff frequency for the notch filter
W = 11   # Width of the filter

# Create a 2D filter to remove specific frequencies
notch_filter = np.ones((rows, cols), np.uint8)

# Set filter to zero in the frequency domain for specific frequency regions
notch_filter[crow-D0-W:crow-D0+W, ccol-D0-W:ccol-D0+W] = 0
notch_filter[crow+D0-W:crow+D0+W, ccol+D0-W:ccol+D0+W] = 0

# Apply the filter in the frequency domain
filtered_f_transform_shifted = f_transform_shifted * notch_filter

# Inverse Fourier transform back to the spatial domain
filtered_image = np.fft.ifftshift(filtered_f_transform_shifted)
filtered_image = np.fft.ifft2(filtered_image)
filtered_image = np.abs(filtered_image).astype(np.uint8)
```



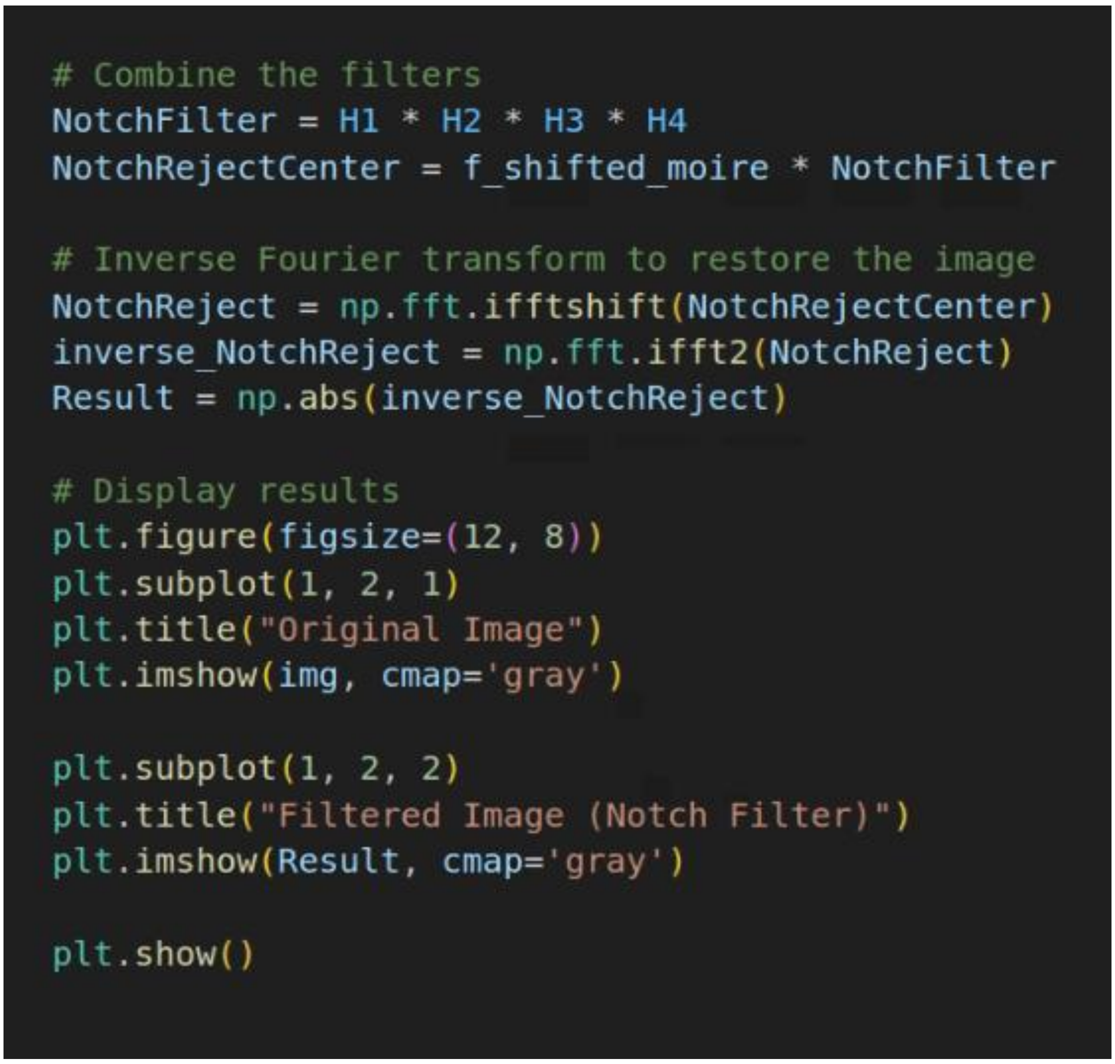
```python
# Calculate the magnitude spectrum of the filtered image
filtered_magnitude_spectrum = np.log(np.abs(filtered_f_transform_shifted) + 1)

# Display the original image and the processed results
plt.figure(figsize=(18, 8))
plt.subplot(1, 2, 1)
plt.title("Original Image")
plt.imshow(image, cmap='gray')

# Display the filtered image
plt.subplot(1, 2, 2)
plt.title("Filtered Image (Notch Filter)")
plt.imshow(filtered_image, cmap='gray')

plt.show()
```



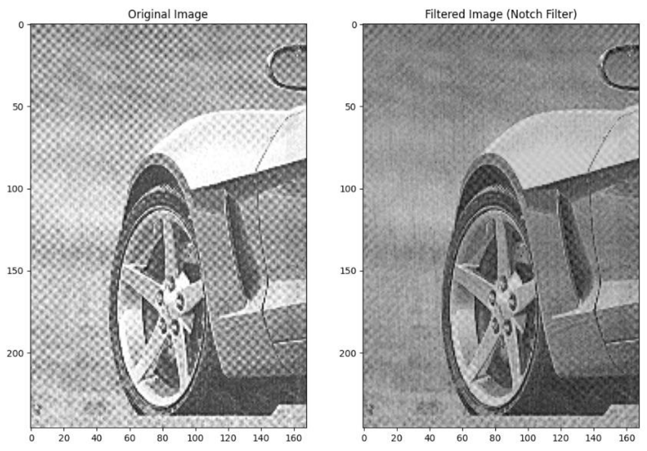
2. Please use FFT and design a frequency filter to cancel the moire pattern noise of the assigned image, ‘car-moire-pattern.tif’, and print out the source code and the processed image? (40)



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

def notch_reject_filter(shape, d0=9, u_k=0, v_k=0):
    P, Q = shape
    H = np.ones((P, Q))
    for u in range(0, P):
        for v in range(0, Q):
            D_uv = np.sqrt((u - P / 2 + u_k) ** 2 + (v - Q / 2 + v_k) ** 2)
            D_muv = np.sqrt((u - P / 2 - u_k) ** 2 + (v - Q / 2 - v_k) ** 2)
            if D_uv <= d0 or D_muv <= d0:
                H[u, v] = 0.0
    return H

# Read the image
img = cv2.imread('car-moire-pattern.tif',0)

# Perform Fourier transform
f_transform_moire = np.fft.fft2(img)
f_shifted_moire = np.fft.fftshift(f_transform_moire)
magnitude_spectrum = 20 * np.log(np.abs(f_shifted_moire))

# Set parameters for multiple filters
img_shape = img.shape
H1 = notch_reject_filter(img_shape, 4, 38, 30)
H2 = notch_reject_filter(img_shape, 4, -42, 27)
H3 = notch_reject_filter(img_shape, 2, 80, 30)
H4 = notch_reject_filter(img_shape, 2, -82, 28)
```



```python
# Combine the filters
NotchFilter = H1 * H2 * H3 * H4
NotchRejectCenter = f_shifted_moire * NotchFilter

# Inverse Fourier transform to restore the image
NotchReject = np.fft.ifftshift(NotchRejectCenter)
inverse_NotchReject = np.fft.ifft2(NotchReject)
Result = np.abs(inverse_NotchReject)

# Display results
plt.figure(figsize=(12, 8))
plt.subplot(1, 2, 1)
plt.title("Original Image")
plt.imshow(img, cmap='gray')

plt.subplot(1, 2, 2)
plt.title("Filtered Image (Notch Filter)")
plt.imshow(Result, cmap='gray')

plt.show()
```



3. Please comment and compare your two design freq. filters? (20)

The two programs both utilize **frequency filtering** techniques, specifically **notch filters**, to suppress unwanted frequency components in the image. However, they differ in design. Here's a comparison and commentary:

1. First Program: Fixed Position Notch Filter

- **Filter Design**: This program uses a rectangular area in the frequency domain to set specific regions to zero, forming a basic notch filter. The filter's center is based on the image's center, applied symmetrically along both the horizontal and vertical directions.
- **Filter Parameters**: The filter center (D0) and width (W) are fixed at 35 and 11, respectively, representing the filter's position and width.
- **Application Scenario**: This design is suitable for suppressing one or a few fixed frequencies, such as periodic interference or specific noise at certain positions.
- **Advantages & Disadvantages**: This design is straightforward and easy to implement but lacks flexibility, making it challenging to fine-tune for multiple frequency locations.

2. Second Program: Dynamic Multi-Frequency Notch Filter

- **Filter Design**: This program uses a **dynamically set** notch_reject_filter function, capable of generating multiple filters based on different frequency centers (u_k, v_k) and radius (d0).
- **Filter Parameters**: Multiple filters target four different frequency locations with two different radii, effectively suppressing several interference sources. Each filter is set according to its frequency center, allowing flexibility in managing multiple noise frequencies.
- **Application Scenario**: This design is suitable for handling images with multiple frequency interferences, such as complex moiré patterns or repetitive noise.
- **Advantages & Disadvantages**: This method is more flexible and can effectively target multi-frequency noise, although it may be

computationally more intensive and complex to implement.

Summary Comparison

- **Filtering Effectiveness**: The first program targets only two symmetrical frequencies, ideal for single-frequency interference. The second program targets four frequencies, allowing broader frequency filtering.
- **Flexibility**: The second program is more flexible, accommodating adjustments for various frequency locations, making it well-suited to complex, multi-frequency noise scenarios.
- **Implementation Difficulty**: The first program is simpler, whereas the second is more complex but provides better results for multi-frequency noise due to its flexible design.