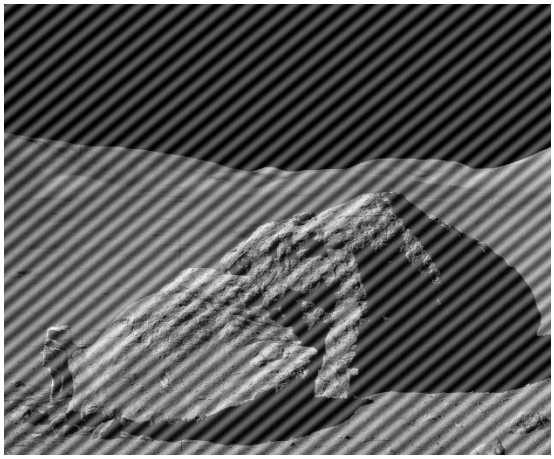
DIP Homework Chapter 4_2

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)

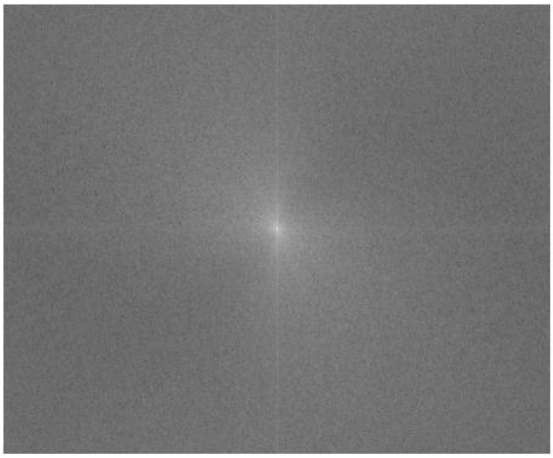
Ans.

0. 原圖(F.1)。

1. 利用 FFT，把照片轉換成 Frequency Domain (F.2)。
2. 發現圖片(F.2)噪點模糊，所以在圖片周圍 Zero Padding (F.3)，使得圖片從原本的 824*1000 擴大至 1648*2000。
3. 將擴大完的圖片經由 FFT，轉換成 Frequency Domain (F.4)。
4. 在圖(F.4)中發現 2 個明顯噪點(774,950), (874,1050)，所以我使用 Gaussian Notch filter 來 Reject 訊號(F.5)。
5. 將過濾後的資料(F.5)，重建至 Spatial Domain，並進行裁切初始 Zero Padding 部分。成果即為(F.6)。

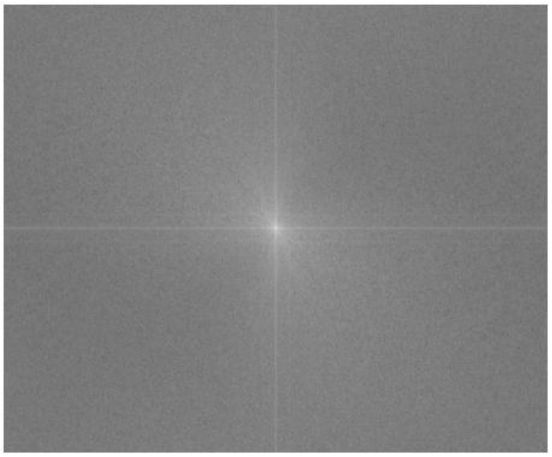


F.1 原始影像圖

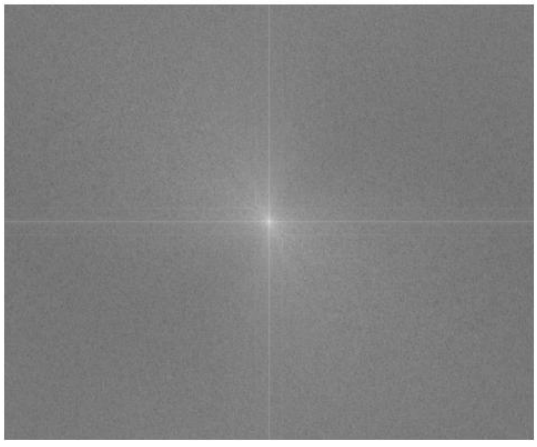


F.2 原始頻譜圖

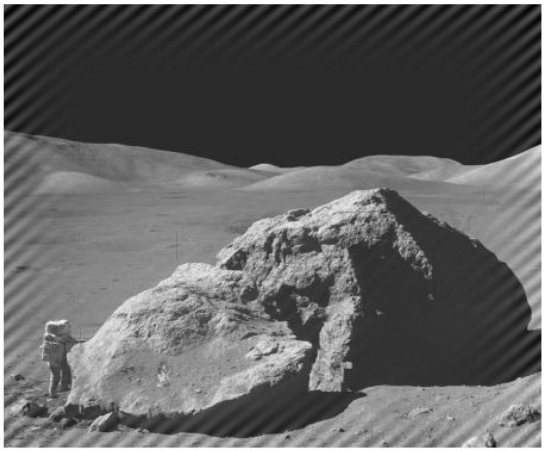
F.3 Zero-Padding 後的空間域圖



F.4 Zero-Padding 後的頻譜圖



F.5 經過 Filter 後的頻譜圖



F.6 Reconstruct 並裁切後的成圖

Source code:

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

# Load the image and convert to grayscale if needed
# ---------------------------
img = plt.imread("astronaut-interference.tif")
rows, cols = img.shape

# Zero-padding to double the size, with centering
padded_rows, padded_cols = rows*2, cols*2
padded_img = np.zeros((padded_rows, padded_cols))

start_r = (padded_rows - rows) // 2
start_c = (padded_cols - cols) // 2
padded_img[start_r:start_r+rows, start_c:start_c+cols] = img

# 2D FFT
F = np.fft.fft2(padded_img)
F_shifted = np.fft.fftshift(F)    # Shift zero-frequency to center
magnitude_spectrum = np.abs(F_shifted)

# 2D Gaussian Notch Filter (two centers)
U, V = np.meshgrid(np.arange(padded_cols), np.arange(padded_rows))
mask = np.ones((padded_rows, padded_cols))    # Initialize mask = 1 (keep all)

# Gaussian notch 1
center1 = (774, 950)      # (row, col)
sigma1 = 5                      # width of notch
D1 = np.sqrt((V - center1[0])**2 + (U - center1[1])**2)
H1 = 1 - np.exp(-(D1**2) / (2*sigma1**2))

# Gaussian notch 2
center2 = (874, 1050)    # (row, col)
sigma2 = 5                      # width of notch
D2 = np.sqrt((V - center2[0])**2 + (U - center2[1])**2)
H2 = 1 - np.exp(-(D2**2) / (2*sigma2**2))

# Combine notches
mask = H1 * H2

# Apply the filter in frequency domain
F_filtered = F_shifted * mask
magnitude_filtered = np.abs(F_filtered)

# Inverse
F_ishift = np.fft.ifftshift(F_filtered)    # Shift back before inverse FFT
img_filtered_full = np.fft.ifft2(F_ishift).real

# Crop the reconstructed image to original size
img_filtered_cropped = img_filtered_full[start_r:start_r+rows, start_c:start_c+cols]
```

```
# Display all images in one interface
fig, axes = plt.subplots(2, 2, figsize=(14, 12), constrained_layout=True)

# Original image
axes[0,0].imshow(img, cmap='gray')
axes[0,0].set_title("Original Image", fontsize=14)
axes[0,0].axis('off')

# Zero-padded image
axes[0,1].imshow(padded_img, cmap='gray')
axes[0,1].set_title("Zero-padded Image (2x)", fontsize=14)
axes[0,1].axis('off')

# Filtered FFT magnitude spectrum
axes[1,0].imshow(np.log(1 + magnitude_filtered), cmap='gray')
axes[1,0].set_title("Filtered FFT Magnitude Spectrum", fontsize=14)
axes[1,0].axis('off')

# Reconstructed image after Gaussian Notch Filter (cropped)
axes[1,1].imshow(img_filtered_cropped, cmap='gray')
axes[1,1].set_title("Reconstructed Image (Cropped to Original Size)", fontsize=14)
axes[1,1].axis('off')

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)

Ans.

0. 原圖(F.7)

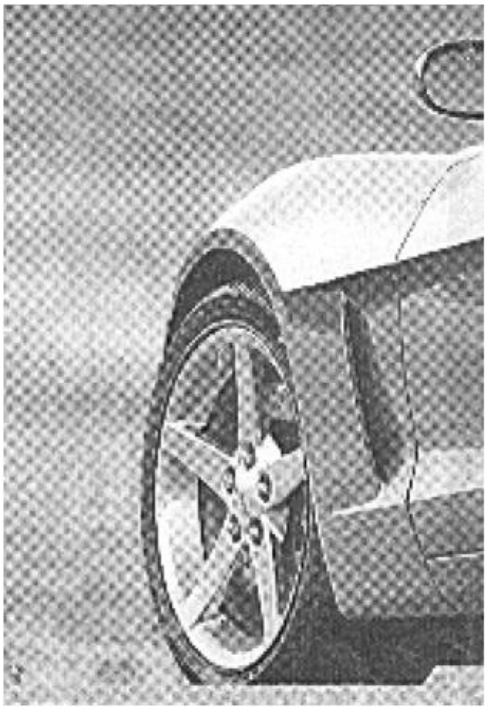
1. 利用 FFT，把照片轉換成 Frequency Domain (F.8)。

2. 在圖(F.8)中發現有 8 個明顯噪點(T.1)，所以我使用 Gaussian Notch filter 來 Reject 訊號(F.9)。

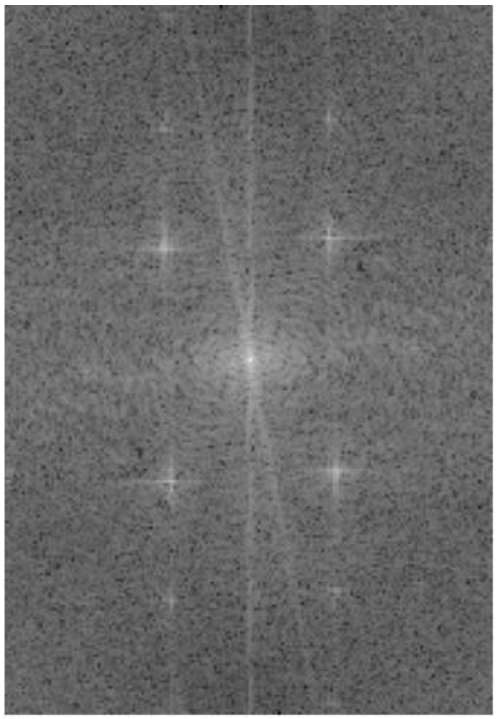
3. 將過濾後的資料(F.9)，重建至 Spatial Domain。成果即為 (F.10)。

(44,54)	(85,55)	(165,57)	(206,57)
(202,114)	(161,113)	(81,111)	(40,111)

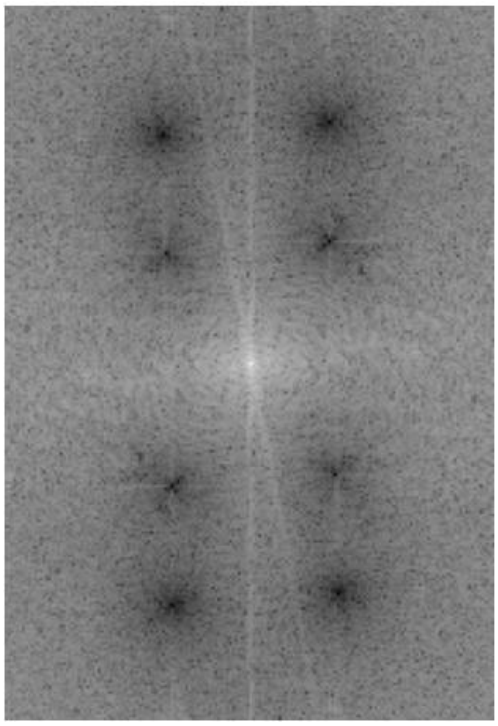
T.1 8 個噪點



F.7 原始影像圖



F.8 原始頻譜圖



F.9 經過 Filter 後的頻譜圖

F.10 Reconstruct 後的成圖

Source code:

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

# Load image and convert to grayscale if needed
img = plt.imread("car-moire-pattern.tif")
rows, cols = img.shape

# 2D FFT
F = np.fft.fft2(img)
F_shifted = np.fft.fftshift(F)    # Shift zero-frequency to center
magnitude_spectrum = np.abs(F_shifted)

# 2D Gaussian Notch Filter (8 centers)
U, V = np.meshgrid(np.arange(cols), np.arange(rows))
mask = np.ones((rows, cols))    # Initialize mask = 1 (keep all)

# Define 8 Gaussian notch centers (row, col) and sigma for each
notch_centers = [(44, 54),(85, 55),(165, 57),(206, 57),(202, 114),(161, 113),(81,
111),(40, 111)]


# Apply Gaussian notch filter + cross reject
for (cy, cx), sigma in zip(notch_centers, sigmas):
      D = np.sqrt((V - cy)**2 + (U - cx)**2)
      H = 1 - np.exp(-(D**2) / (2*sigma**2))
      mask *= H

# Apply mask in frequency domain
F_filtered = F_shifted * mask
magnitude_filtered = np.abs(F_filtered)

# Inverse FFT to reconstruct the image
F_ishift = np.fft.ifftshift(F_filtered)
img_filtered = np.fft.ifft2(F_ishift).real


# Display four images
fig, axes = plt.subplots(2, 2, figsize=(14, 12), constrained_layout=True)
# 1. Original image
axes[0,0].imshow(img, cmap='gray')
axes[0,0].set_title("Original Image", fontsize=14)
axes[0,0].axis('off')
# 2. Original FFT magnitude spectrum
axes[0,1].imshow(np.log(1 + magnitude_spectrum), cmap='gray')
axes[0,1].set_title("Original FFT Magnitude Spectrum", fontsize=14)
axes[0,1].axis('off')
# 3. Filtered FFT magnitude spectrum
axes[1,0].imshow(np.log(1 + magnitude_filtered), cmap='gray')
axes[1,0].set_title("Filtered FFT Magnitude Spectrum", fontsize=14)
axes[1,0].axis('off')
# 4. Reconstructed image
axes[1,1].imshow(img_filtered, cmap='gray')
axes[1,1].set_title("Reconstructed Image after Notch Filters", fontsize=14)
axes[1,1].axis('off')
plt.show()
```

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

Ans.

兩個設計都使用 **Gaussian Notch Filter(G.N.F.)**而非理想 notch，**G.N.F 在頻域邊界平滑，可以減少 Ringing 產生**，並且可調整標準差 σ 來抑制噪點。兩者處理流程也相同：**對圖像進行 FFT，計算 Notch 距離 D，用高斯函數濾波器，將多個 notch mask 相乘後用於頻域，再進行 IFFT 回到空間域重建圖像**。

兩個設計的差異在於**干擾頻率的數量與影響範圍**。**第一個設計只針對 2 個噪點，並使用 Zero-Padding 放大頻域解析度**，使高斯 notch 更精準，對單一或少量干擾效果明顯且圖像細節保留較好；**第二個設計則針對 8 個干擾點，濾除多個噪點，沒有使用 Zero-Padding**，降低 FFT 的計算量，但可能使圖像某些細節略微模糊。