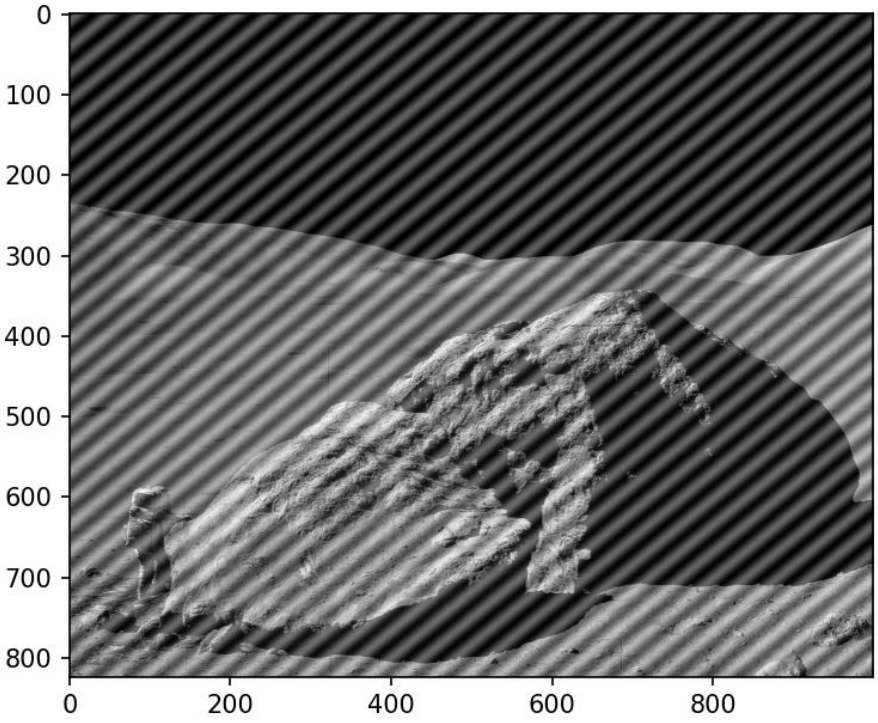
數位影像處理 DIP Homework Chapter 4_2

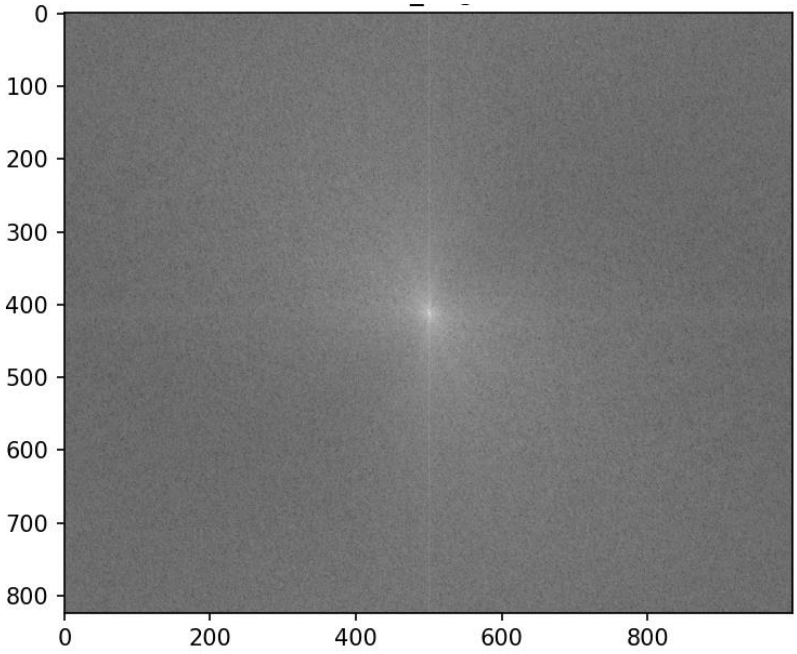
1. Please use FFT and design a **frequency filter** to cancel the sinusoidal noise of the assigned image, 'astronaut-interference.tif', and print out the source code and the processed image? (40)

原圖




```
# matplotlib read image
img = mpimg.imread("astronaut-interference.tif")
print(img.shape)
# Output Images
plt.imshow(img, cmap=cm.gray)
plt.show()
```

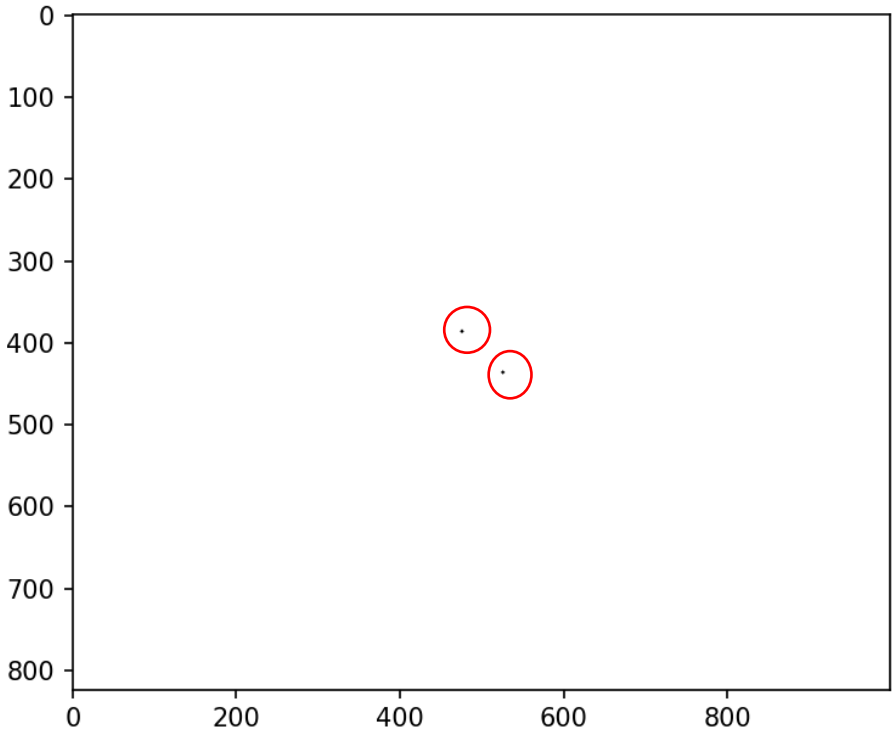
原圖經 Discrete Fourier Transform (FFT_image)




```
# Fourier Transform
fft_img = np.fft.fftshift(np.fft.fft2(img))
plt.imshow(np.log(abs(fft_img) + 1), cmap='gray'), plt.title("fft_img")
plt.show()
```

designed mask

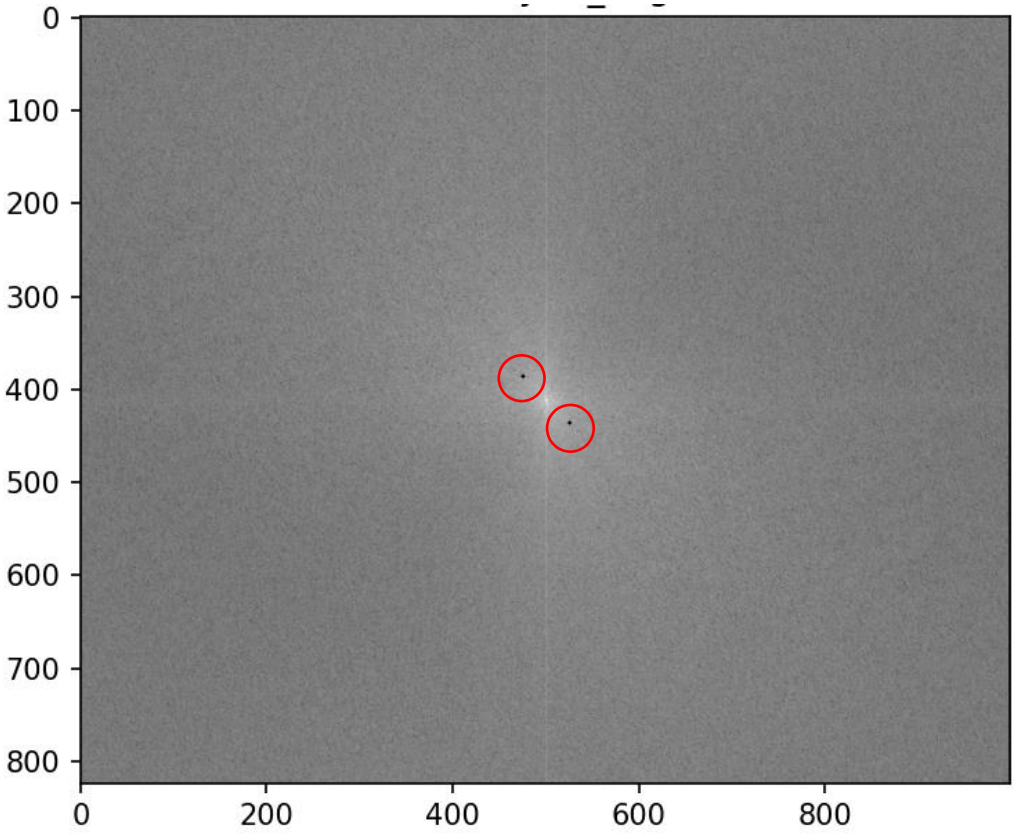
(這邊以紅圓圈標記改動位置，紅圈標記的那兩點像素設為 0)



```
## modify on FFT image
mask = np.zeros_like(img)
mask = cv2.circle(mask, (475,386), 2, (255,255,255), -1)  # 左上角 burst
mask = cv2.circle(mask, (525,436), 2, (255,255,255), -1)  # 右下角 burst
mask = 255 - mask
mask = mask/mask.max()
plt.imshow(mask, cmap='gray'), plt.title("mask")
plt.show()
```

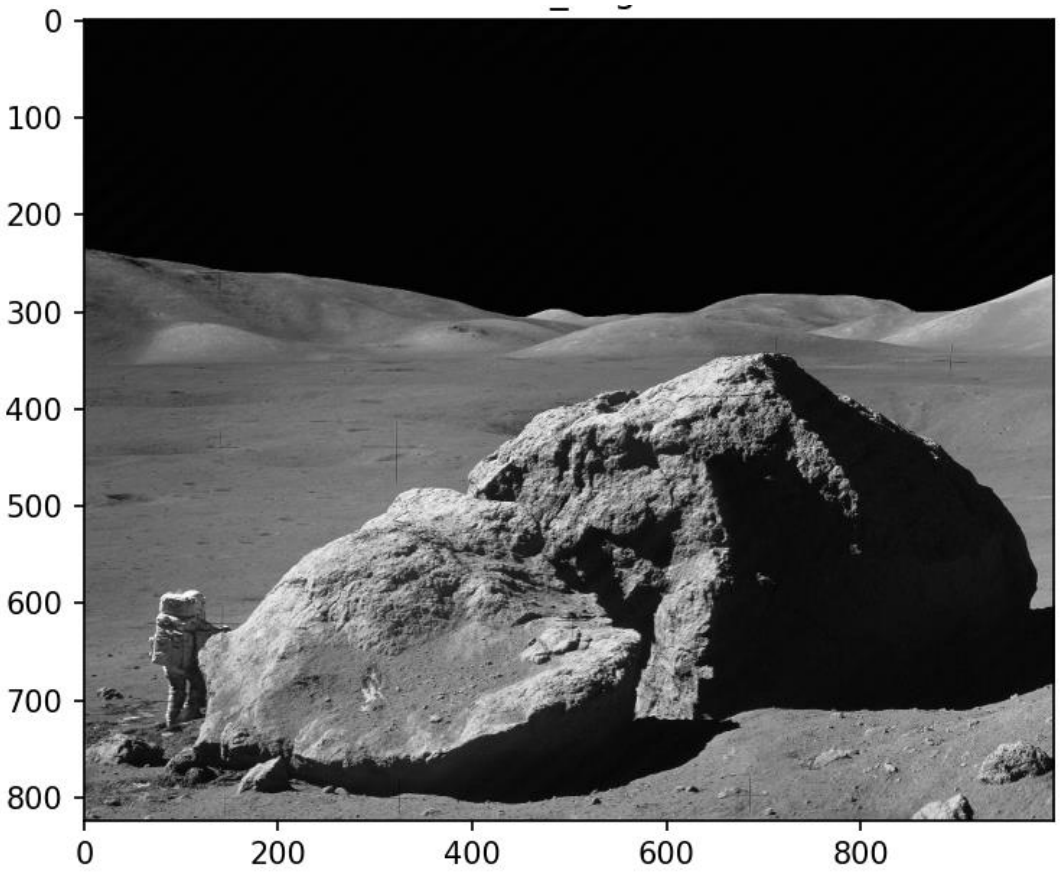
FFT_image * designed mask

(這邊以紅圓圈標記改動位置)




```
modify_fft_img = fft_img*mask
plt.imshow(np.log(abs(modify_fft_img) + 1), cmap='gray'), plt.title("modify fft_img")
plt.show()
```

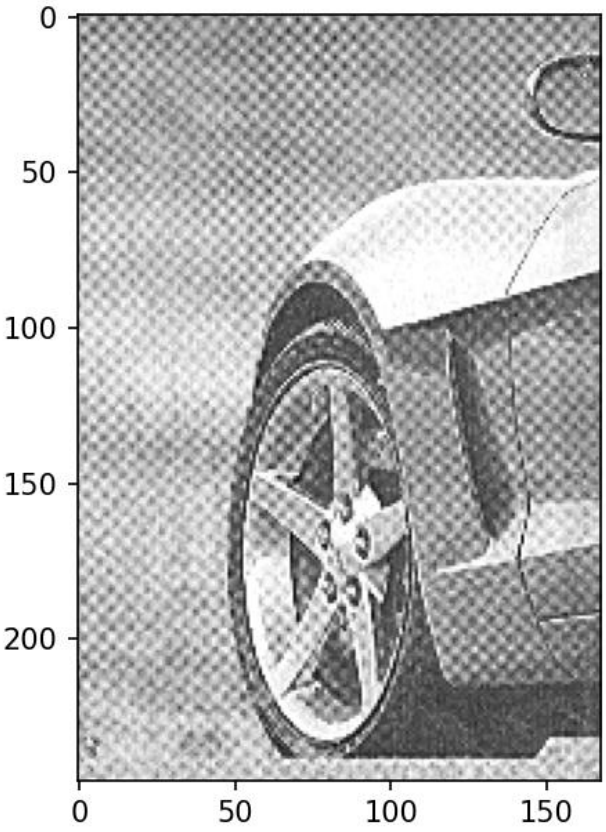
FFT_image 經 Inverse Discrete Fourier Transform (IFFT_image)




```
## inverse Fourier Transform
ifft_img = np.fft.ifft2(np.fft.ifftshift(modify_fft_img))
print(f"ifft_img.max(), ifft_img.min() = {ifft_img.max(), ifft_img.min()}")
plt.imshow(np.abs(ifft_img), cmap='gray'), plt.title("ifft_img")
plt.show()
```

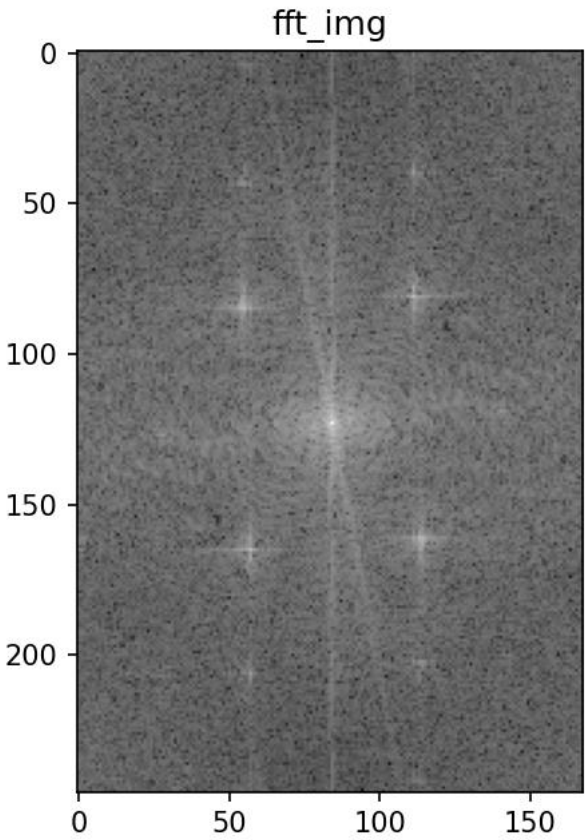
2. Please use FFT and design a **frequency filter** to cancel the moire-pattern noise of the assigned image, 'car-moire-pattern.tif', and print out the source code and the processed image? (40)

原圖



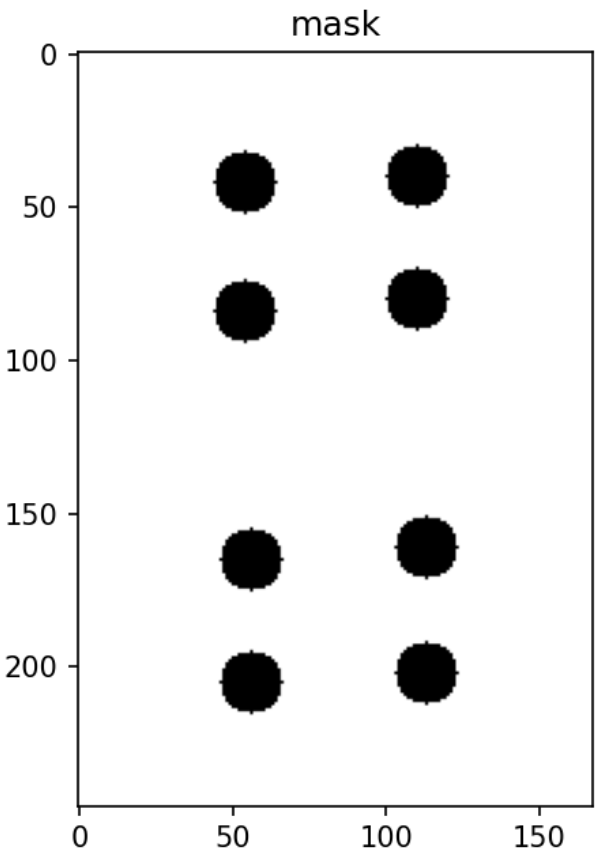

```
# matplotlib read image
img = mpimg.imread("car-moire-pattern.tif")
print(img.shape)
# Output Images
plt.imshow(img, cmap=cm.gray)
plt.show()
```

原圖經 Discrete Fourier Transform (FFT_image)



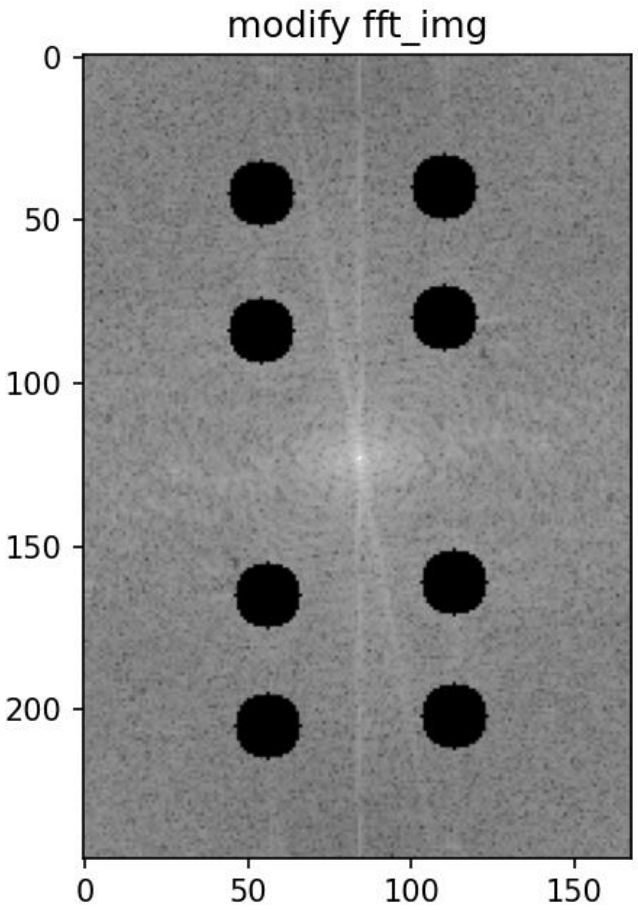

```
# Fourier Transform
fft_img = np.fft.fftshift(np.fft.fft2(img))
plt.imshow(np.log(abs(fft_img) + 1), cmap='gray'), plt.title("fft_img")
plt.show()
```

designed mask



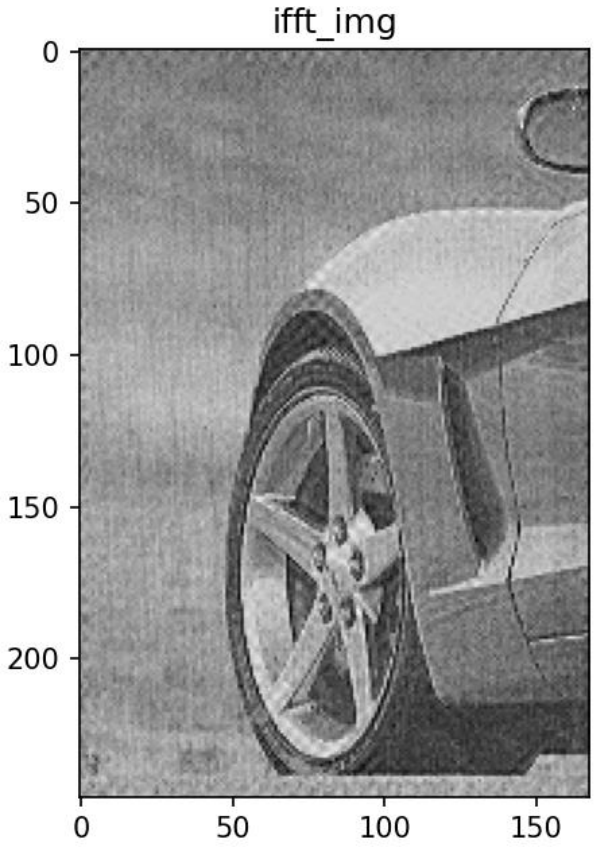

```
## modify on FFT image
mask = np.zeros_like(img)
# 左邊那排
mask = cv2.circle(mask, (54,42), 10, (255,255,255), -1)
mask = cv2.circle(mask, (54,84), 10, (255,255,255), -1)
mask = cv2.circle(mask, (56,165), 10, (255,255,255), -1)
mask = cv2.circle(mask, (56,205), 10, (255,255,255), -1)
# 右邊那排
mask = cv2.circle(mask, (110,40), 10, (255,255,255), -1)
mask = cv2.circle(mask, (110,80), 10, (255,255,255), -1)
mask = cv2.circle(mask, (113,161), 10, (255,255,255), -1)
mask = cv2.circle(mask, (113,202), 10, (255,255,255), -1)
mask = 255 - mask
mask = mask/mask.max()
plt.imshow(mask, cmap='gray'), plt.title("mask")
plt.show()
```

FFT_image * designed mask



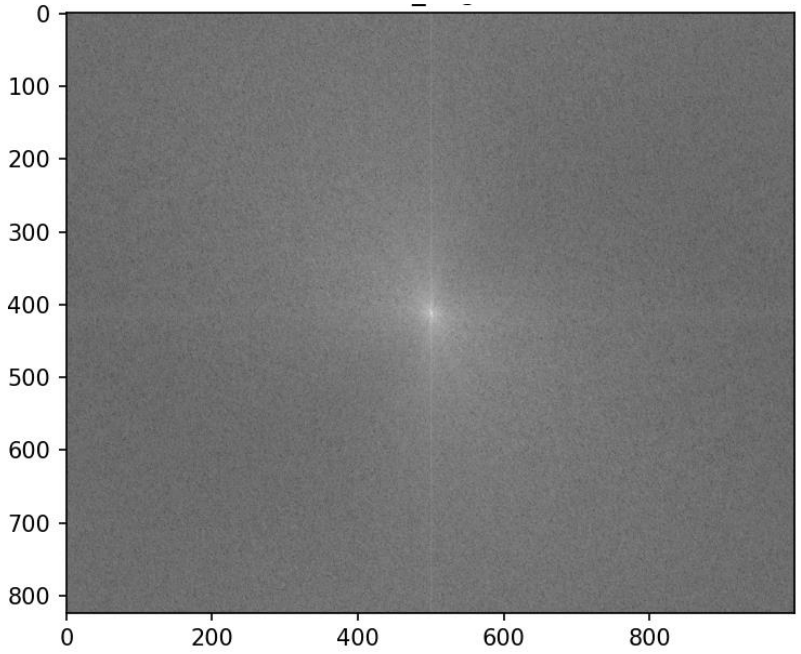

```
modify_fft_img = fft_img*mask
plt.imshow(np.log(abs(modify_fft_img) + 1), cmap='gray'), plt.title("modify fft_img")
plt.show()
```

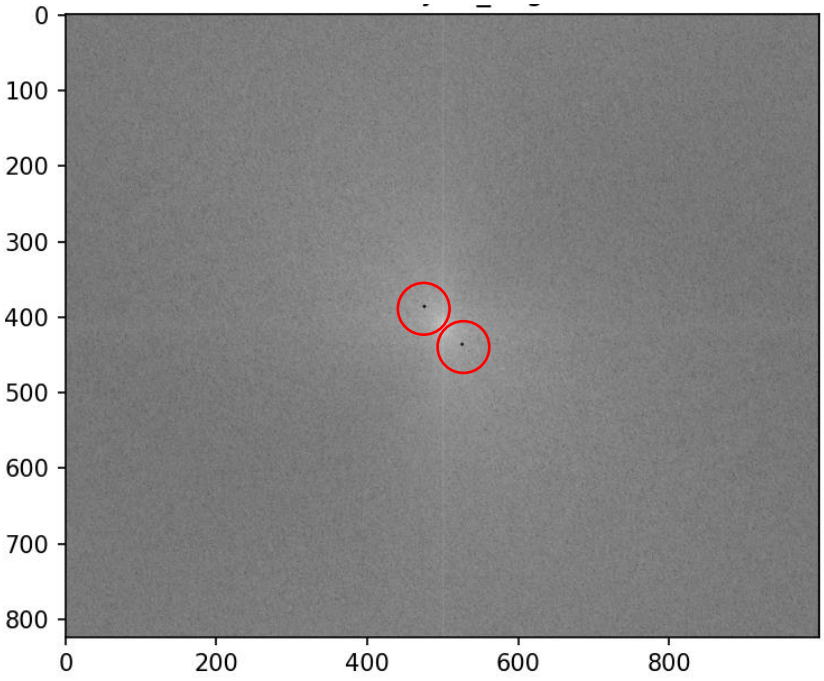
FFT_image 經 Inverse Discrete Fourier Transform (IFFT_image)

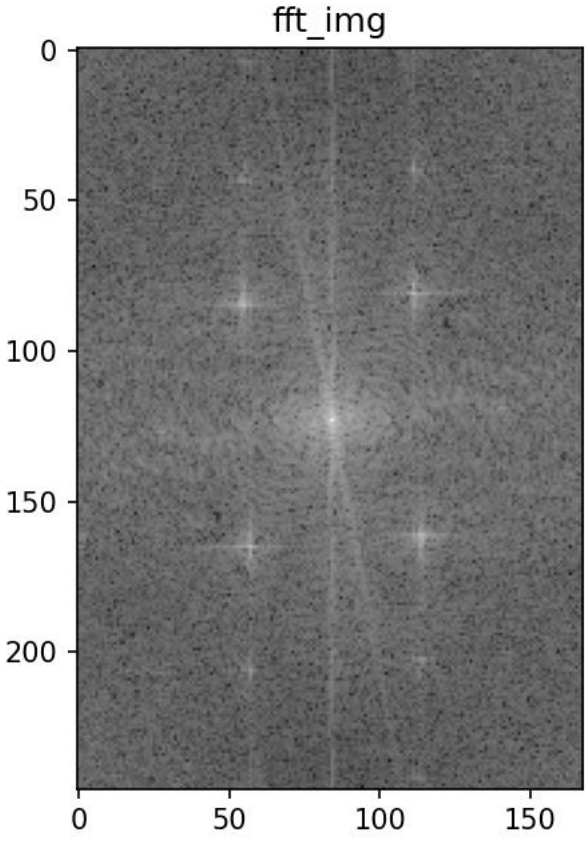


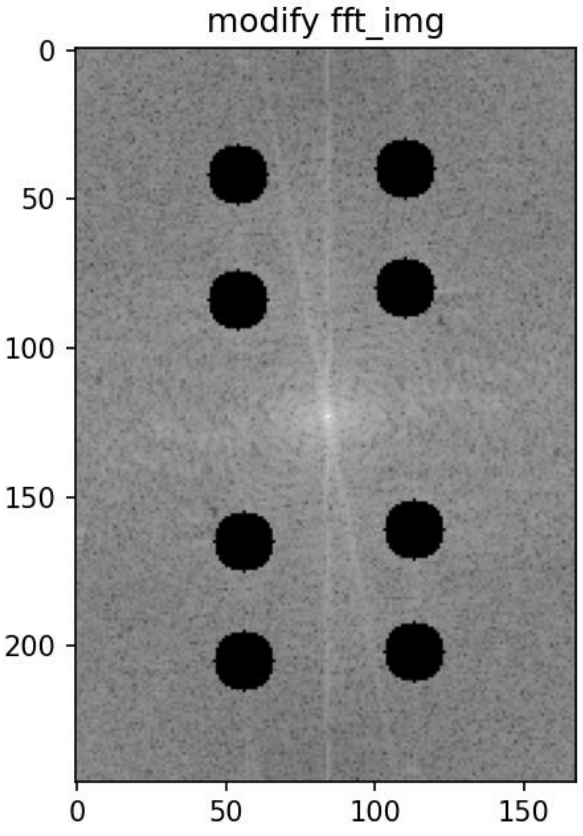

```
## inverse Fourier Transform
ifft_img = np.fft.ifft2(np.fft.ifftshift(modify_fft_img))
print(f"ifft_img.max(), ifft_img.min() = {ifft_img.max(), ifft_img.min()}")
plt.imshow(np.abs(ifft_img), cmap='gray'), plt.title("ifft_img")
plt.show()
```

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

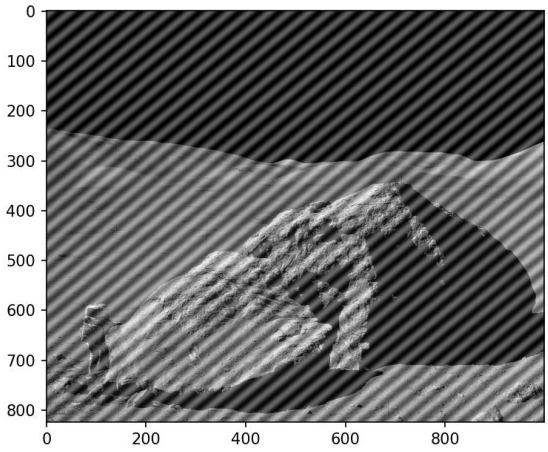




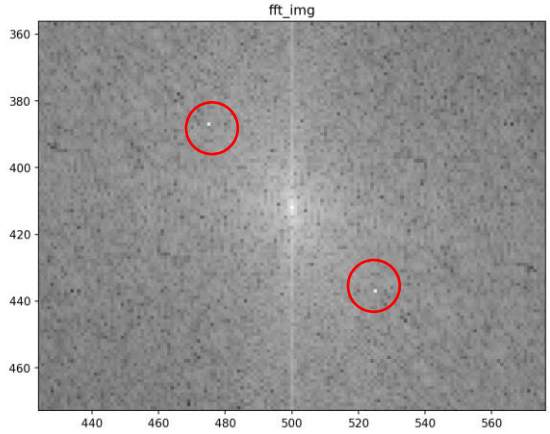

fft_img
0
50
100
150
200
0
50
100
150

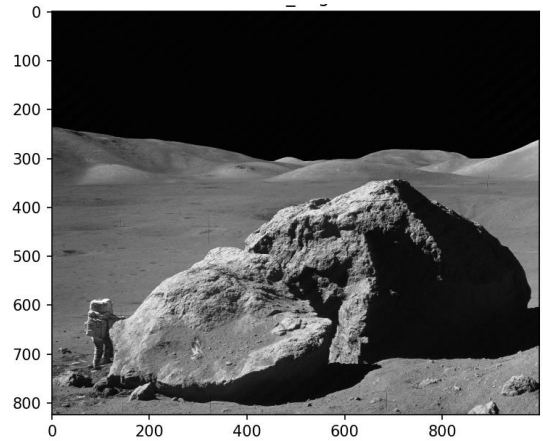

modify fft_img
0
50
100
150
200
0
50
100
150

上下兩張圖分別為修改前與修改後的 frequency filter, 針對第一題的原圖

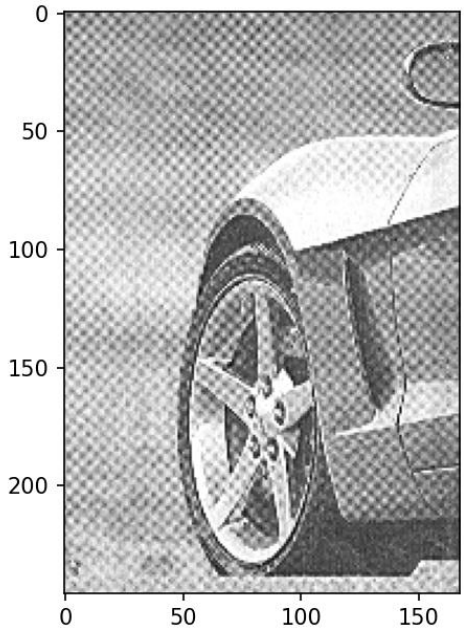


由於圖像中有特定的低頻雜訊，因此在設計 frequency filter 時，將在特定頻率上的兩個亮點像素值設為 0 (需放大較能清楚看見亮點)，以除掉該特定頻率的訊號，再 inverse fourier transform 回去得結果圖，由結果圖可見，雜訊確實由此兩再頻域的亮點產生。

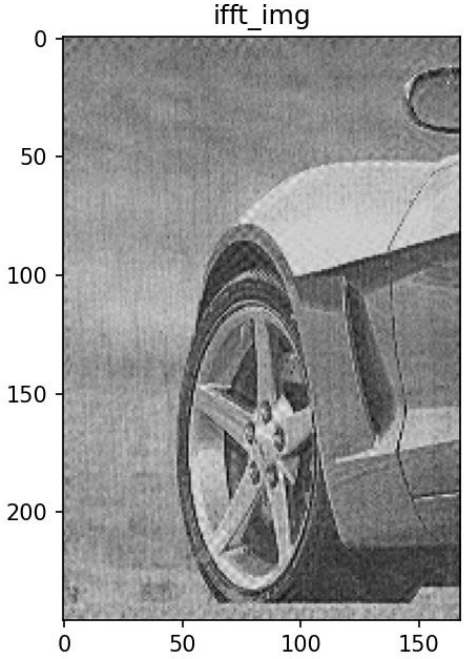




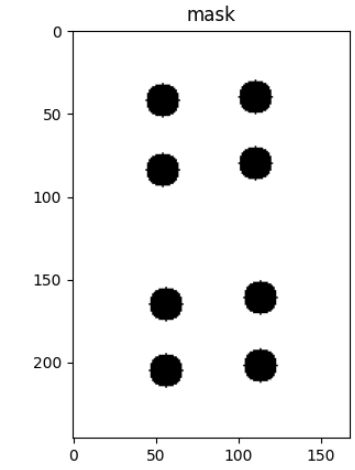
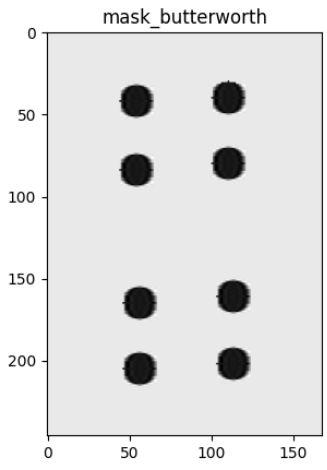
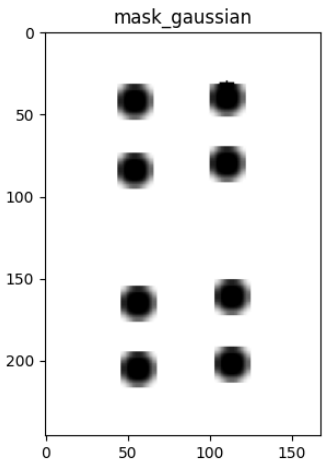
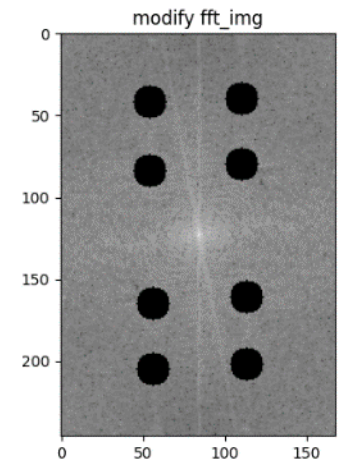
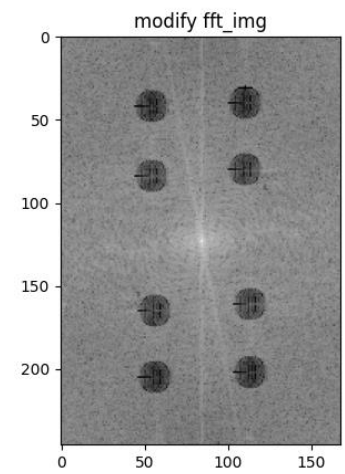
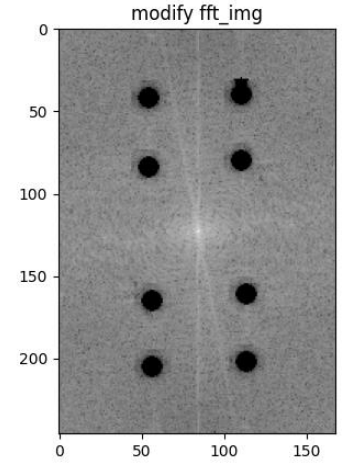
而第二題的原圖



雖然 moire pattern 的雜訊非高頻也非低頻(藉在中間)，但仍可找尋造成圖片中雜訊的頻率點，並如同處理圖一的方式，使用 mask 將其像素遮掉，或在該點套上 low pass filter，而本次我是使用 ideal low pass filter, 最終得到如下的解果圖:



另附上以 butterworth 及 gaussian filter 實作的結果，但個人覺得沒有差很多。

Ideal mask	Butterworth mask	Gaussian mask
		
Ideal **fft** image	Butterworth **fft** image	Gaussian **fft** image
		
Ideal **ifft** image	Butterworth **ifft** image	Gaussian **ifft** image