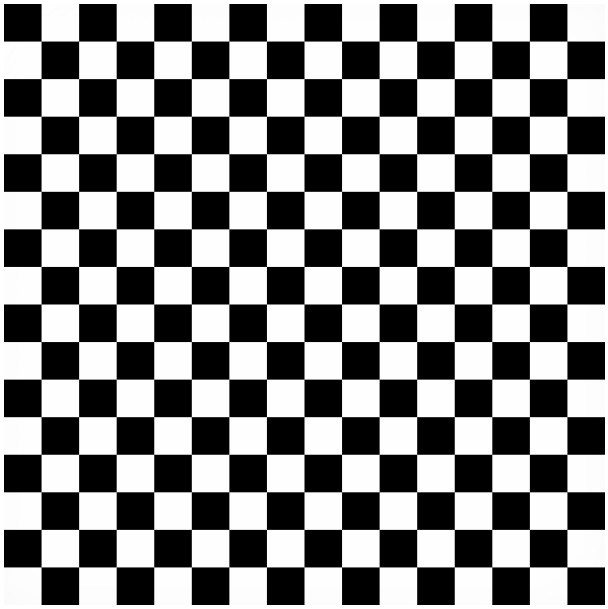
電信碩 309513061 簡詠倫 HW3_3

1. 參考了課本，發現圖片方格長度為課本的一半，因此我設計的低通濾波器 kernel size 為 257*257、K=1、σ =64，下圖為結果以及程式碼，看起來成果還不差。



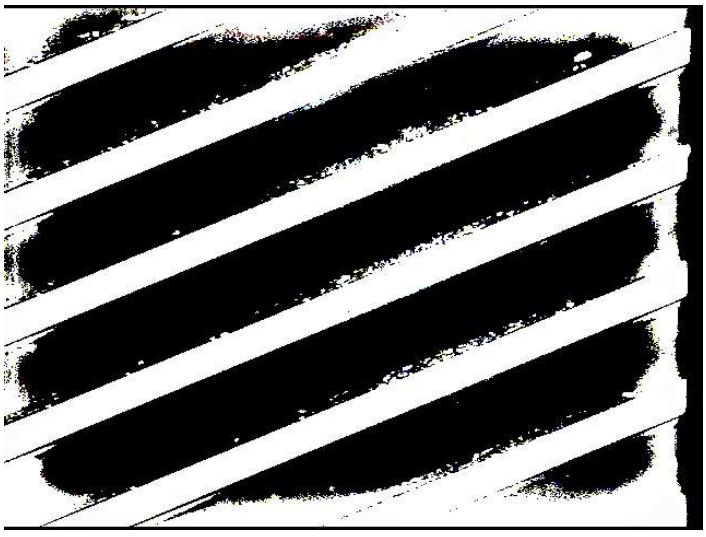
```python
import numpy as np
import matplotlib.pyplot as plt
import cv2
img = cv2.imread('checkerboard1024-shaded.tif')

def gaussain_kernel(ker_size,center_value,var_value):
    gau = np.zeros([ker_size,ker_size])
    for s in range(ker_size):
        for t in range(ker_size):
            gau[s,t] = np.exp(-(pow(s-center_value,2)+pow(t-center_value,2))/(2*var_value))
    k = np.sum(gau)
    gau = gau/k
    return gau

def lpf(img,gau,center_value,img_pad):
    result = np.zeros_like(img)
    noise = np.zeros_like(img)
    for k in range(img.shape[2]):
        for i in range(center_value,center_value+img.shape[0]):
            for j in range(center_value,center_value+img.shape[1]):
                noise[i-center_value,j-center_value,k] = \
                np.sum(img_pad[i-center_value:i+center_value+1,j-center_value:j+center_value+1,k]*gau)
                if noise[i-center_value,j-center_value,k] != 0:
                    result[i-center_value,j-center_value,k] = \
                    int(img[i-center_value,j-center_value,k]/noise[i-center_value,j-center_value,k])*255
                else:
                    result[i-center_value,j-center_value,k] = img[i-center_value,j-center_value,k]
    return result,noise

kersize = 257
var = 64*64
center = kersize//2
gaussian = np.zeros([kersize,kersize])
padding = cv2.copyMakeBorder(img,center,center,center,center,cv2.BORDER_CONSTANT,value=(0,0,0,0))
gau_kernel = gaussain_kernel(kersize,center,var)
result_img,result_noise = lpf(img,gau_kernel,center,padding)
cv2.imwrite('board_result.jpg',result_img)
cv2.imwrite('noise_result.jpg',result_noise)
```

2. 由於此圖較小，我嘗試使用較小的 kernel，因此 kernel size 改為 129*129，K=1、σ=64。以下圖為結果以及程式碼。



```python
import numpy as np
import matplotlib.pyplot as plt
import cv2
img = cv2.imread('N1.bmp')

def gaussain_kernel(ker_size,center_value,var_value):
    gau = np.zeros([ker_size,ker_size])
    for s in range(ker_size):
        for t in range(ker_size):
            gau[s,t] = np.exp(-(pow(s-center_value,2)+pow(t-center_value,2))/(2*var_value))
    k = np.sum(gau)
    gau = gau/k
    return gau

def lpf(img,gau,center_value,img_pad):
    result = np.zeros_like(img)
    noise = np.zeros_like(img)
    for k in range(img.shape[2]):
        for i in range(center_value,center_value+img.shape[0]):
            for j in range(center_value,center_value+img.shape[1]):
                noise[i-center_value,j-center_value,k] = \
                np.sum(img_pad[i-center_value:i+center_value+1,j-center_value:j+center_value+1,k]*gau)
                if noise[i-center_value,j-center_value,k] != 0:
                    result[i-center_value,j-center_value,k] = \
                    int(img[i-center_value,j-center_value,k]/noise[i-center_value,j-center_value,k])*255
                else:
                    result[i-center_value,j-center_value,k] = img[i-center_value,j-center_value,k]
    return result,noise

kersize = 129
var = 64*64
center = kersize//2
gaussian = np.zeros([kersize,kersize])
padding = cv2.copyMakeBorder(img,center,center,center,center,cv2.BORDER_CONSTANT,value=(0,0,0,0))
gau_kernel = gaussain_kernel(kersize,center,var)
result_img,result_noise = lpf(img,gau_kernel,center,padding)
cv2.imwrite('n1_result.jpg',result_img)
cv2.imwrite('n1_noise_result.jpg',result_noise)
```

3. 不同的參數決定了高斯核的形狀，核的大小與影像中物件大小的關係可導致空間濾波的成效有無。在第一題中的核與方形的大小關係得以使陰影改善，但在第二題變得像是對影像門檻化的結果。若能更了解影像中物件大小來設計高斯核的參數，則能對影像濾波處理有更好的改善，這是未來需要改善的課題。