

影像處理 HW3-3

312512049 電控碩 顏志憲

1. Self-designed Lowpass Gaussian Filter Kernels:

```
import cv2
import numpy as np
import os

kernel_size_1 = 250
kernel_size_2 = 120
std_1 = 64
std_2 = 512

def gaussian_kernel(size, std):
    kernel = np.zeros([size, size])
    center = size // 2
    for i in range(size):
        for j in range(size):
            kernel[i, j] = np.exp(-((i - center)**2 + (j - center)**2) / (2 * std**2))
    return kernel / np.sum(kernel)

def low_pass_filter(img, kernel, img_pad):
    noise = cv2.filter2D(img_pad, -1, kernel)
    noise = noise[kernel.shape[0] // 2 : kernel.shape[0] // 2 + img.shape[0] ,
                kernel.shape[1] // 2 : kernel.shape[1] // 2 + img.shape[1]]
    noise[noise == 0] = 1
    img_filtered = (img / noise) * 255 / 2
    return img_filtered, noise

if __name__ == '__main__':
    img_1 = cv2.imread('checkerboard1024-shaded.tif')
    img_2 = cv2.imread('N1.bmp')

    img_pad1 = cv2.copyMakeBorder(img_1, kernel_size_1 // 2, kernel_size_1 // 2,
                                   kernel_size_1 // 2, kernel_size_1 // 2, cv2.BORDER_CONSTANT, value=(0, 0, 0))
    img_pad2 = cv2.copyMakeBorder(img_2, kernel_size_2 // 2, kernel_size_2 // 2,
                                   kernel_size_2 // 2, kernel_size_2 // 2, cv2.BORDER_CONSTANT, value=(0, 0, 0))

    kernel_1 = gaussian_kernel(kernel_size_1, std_1)
    kernel_2 = gaussian_kernel(kernel_size_2, std_2)

    img_filtered_1, noise_1 = low_pass_filter(img_1, kernel_1, img_pad1)
    img_filtered_2, noise_2 = low_pass_filter(img_2, kernel_2, img_pad2)

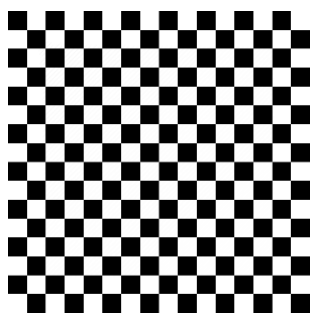
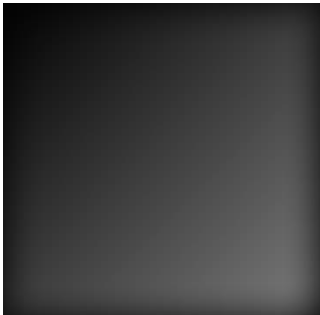
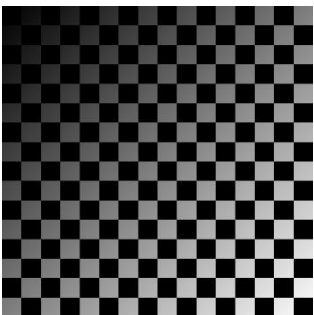
    output_dir = 'img1_output'
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    cv2.imwrite(os.path.join(output_dir, 'original_image_1.png'), img_1)
    cv2.imwrite(os.path.join(output_dir, 'filtered_image_1.png'), img_filtered_1)
    cv2.imwrite(os.path.join(output_dir, 'noise_image_1.png'), noise_1)
    output_dir = 'img2_output'
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)
    cv2.imwrite(os.path.join(output_dir, 'original_image_2.png'), img_2)
    cv2.imwrite(os.path.join(output_dir, 'filtered_image_2.png'), img_filtered_2)
    cv2.imwrite(os.path.join(output_dir, 'noise_image_2.png'), noise_2)
```

整個做法如下:讀取圖片後,首先對圖像進行填充,以便後續的卷積操作。
隨後定義要進行卷積的 Lowpass Gaussian filter Kernel,最後進行低通高斯濾波,

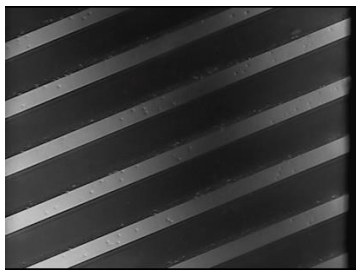
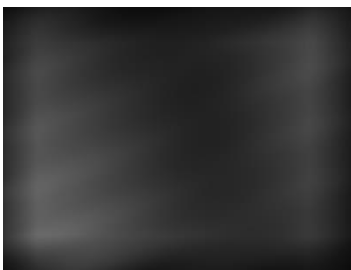
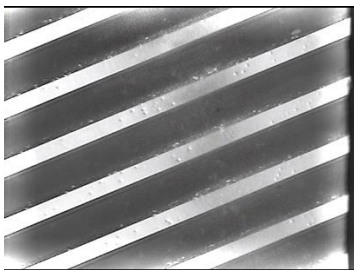
詳細方法使用 OpenCV 中的 `cv2.filter2D` 來對圖像進行卷積操作，得到圖片中的陰影噪聲，接著將噪聲剪裁為原本圖像大小後，將影像除以該噪聲來將原始影像中的噪聲濾除，最後利用一個係數來調整影像的強度。

去噪結果：

1. Checkerboard1024-shaded.tif:

Origin	Shading noise	Final
		

2. N1.bmp:

Origin	Shading noise	Final
		

2. Comment:

由於圖片中噪聲的嚴重程度不同，以及圖片大小不同，因此不同的 Kernel 參數也有不同的結果，利用 kernel size 越大越能夠去除原影像的細節，而方差越小也能夠保留越詳細的原圖特徵。由兩格結果對比可以發現，圖一的去噪效果較好，我認為是因為噪聲是遞減的形式，而圖二事先增在減，造成在平滑處理圖像獲得噪聲時，獲得較模糊的噪聲，因此第二張圖的結果去除了掉了一些原本噪聲較大的細節。