Lab03 - 107061123 孫元駿

Proj04-01

Explanation

在這題我主要實作了兩個 function，分別是 myDFT2 和 myIDFT2。
我在 myDFT2 中用了兩種方式去實作，第一種是用 for loop 去刻課本上的公式。

```
tmp = inputImage(x,y)*exp(-2j*pi*((u-1)*(x-1)/M+(v-1)*(y-1)/N));
dft2Image(u,v) = dft2Image(u,v) + tmp;
```

另外一個方式，我將二維傅立葉轉換公式拆解成兩個一維的傅立葉轉換公式，這樣的方式可以加快運算的速度。

```
rowMatrix = exp(-2j * pi * ((0:M-1)' * (0:M-1)) / M);
colMatrix = exp(-2j * pi * ((0:N-1)' * (0:N-1)) / N);
dft2Image = rowMatrix * inputImage * colMatrix;
```

在 myIDFT2 的 function 中，我也做了兩種方式。第一種是用 for loop 去刻課本上的公式。

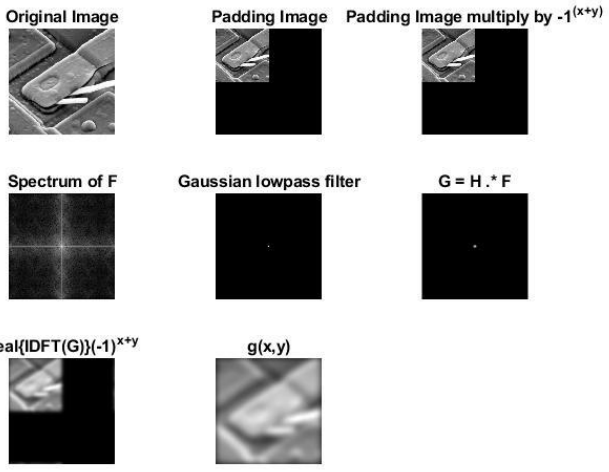
```
tmp = inputImage(x,y)*exp(2j*pi*((u-1)*(x-1)/w+(v-1)*(y-1)/h));
idft2Image(u,v) = idft2Image(u,v) + tmp;
```

另外一個方式，我將二維傅立葉轉換公式拆解成兩個一維的傅立葉轉換公式，這樣的方式可以加快運算的速度。

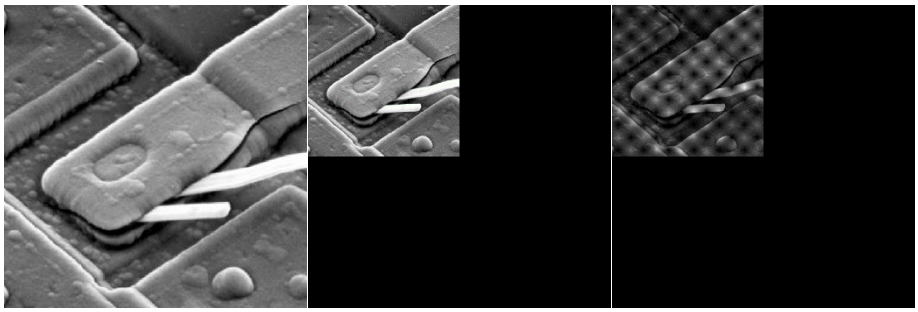
```
rowMatrix = exp(-2j * pi * ((0:M-1)' * (0:M-1)) / M);
colMatrix = exp(-2j * pi * ((0:N-1)' * (0:N-1)) / N);
idft2Image = rowMatrix' * (inputImage / (M * N)) * colMatrix';
```

這樣的方式和上面的 myDFT2 的效果一樣，可以加速運算。

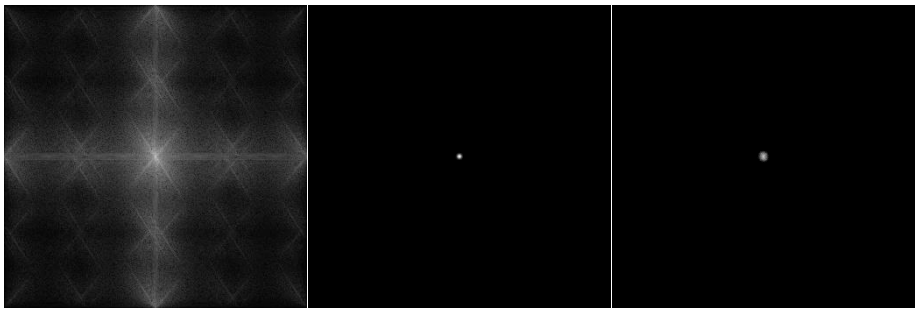
Results



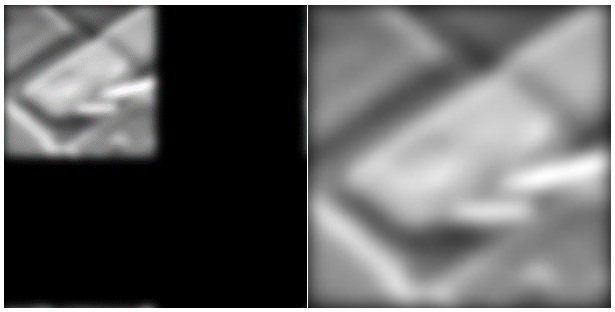
(a) / (b) / (c)



(d) / (e) / (f)



(g) / (h)



Discussion

可以將原本的二維傅立葉轉換公式拆解成兩個一維的傅立葉轉換公式，分別針對 Row 和 Column 產生一個矩陣，再利用矩陣的乘法對原圖進行相乘。這樣可以加快運算的效率，從手刻的 for loop 變成運算速度較快的 matrix multiplication。

Proj04-02

Explanation

在這題主要需要實作的函式是 myFftshift()。我們可以利用下面這張圖去實作。

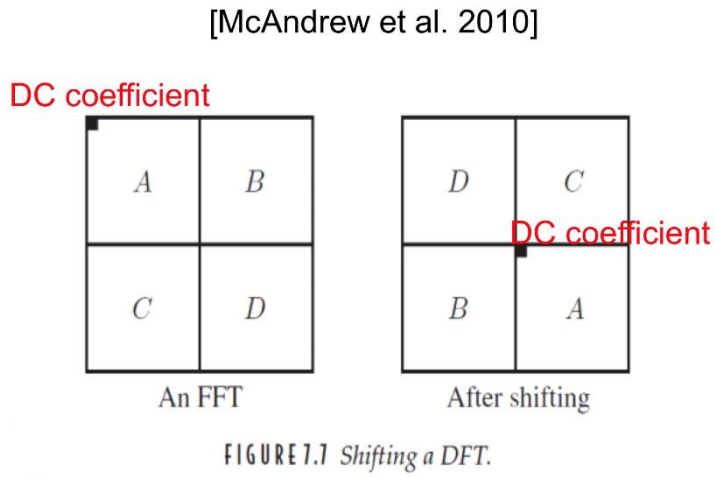
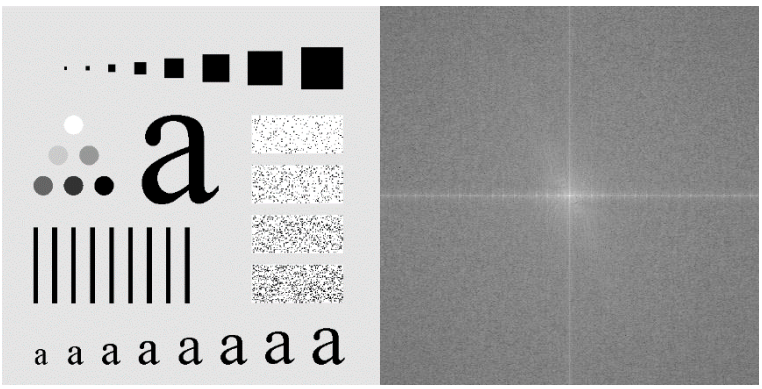


FIGURE 7.7 *Shifting a DFT.*

```
% D -> A
shiftImage(1:w/2,1:h/2) = inputImage(w/2+1:w,h/2+1:h);
% C -> B
shiftImage(w/2+1:w, 1:h/2) = inputImage(1:w/2, h/2+1:h);
% B -> C
shiftImage(1:w/2, h/2+1:h) = inputImage(w/2+1:w, 1:h/2);
% A -> D
shiftImage(w/2+1:w, h/2+1:h) = inputImage(1:w/2, 1:h/2);
```

Results

Spatial Domain / Frequency Domain



Discussion

直接從圖片計算的 mean 是 207.3147，經過傅立葉轉換後的 Center frequency component 也是 207.3147。兩者會相同是因為在 frequency domain 中的中心點的值會是最低的頻率，剛好會是 spatial domain 中 pixel 的平均值。可以從公式中推得這個結論。

$$F(0,0) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) = \frac{1}{MN} \bar{f}(x,y)$$

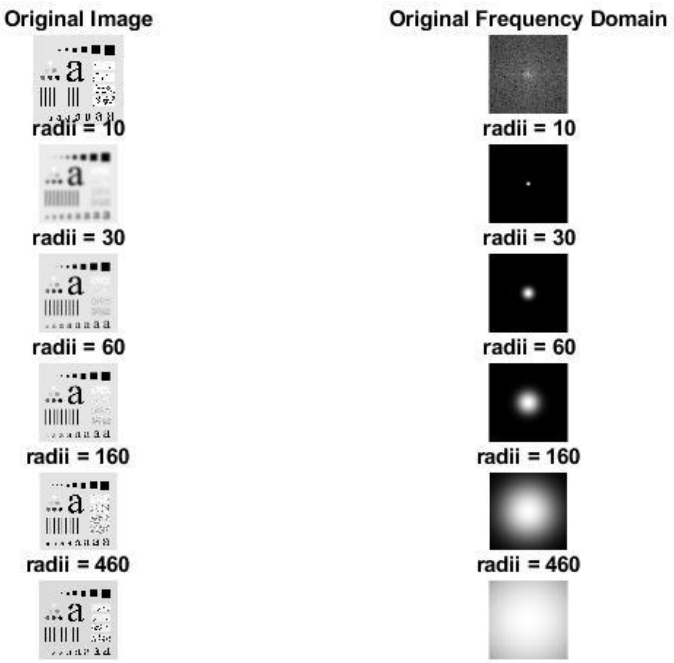
Proj04-03

Explanation

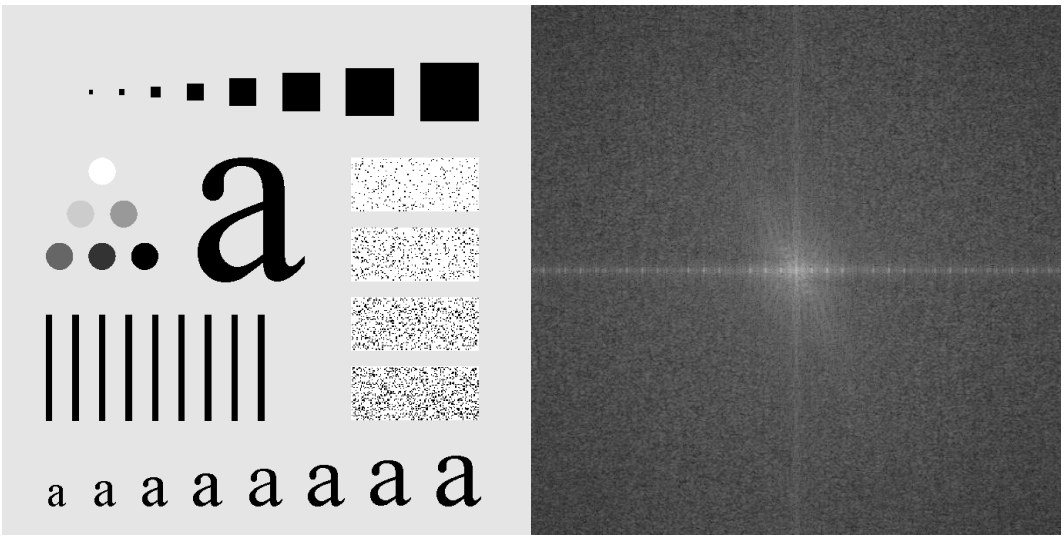
在 myGLPF(D0,M,N)中，我利用 for loop 的方式去時做出大小為 M*N 的 Gaussian lowpass filter。

```
H(i,j)= exp(-((i-M/2)^2+(j-N/2)^2)/(2*(D0^2)));
```

Results

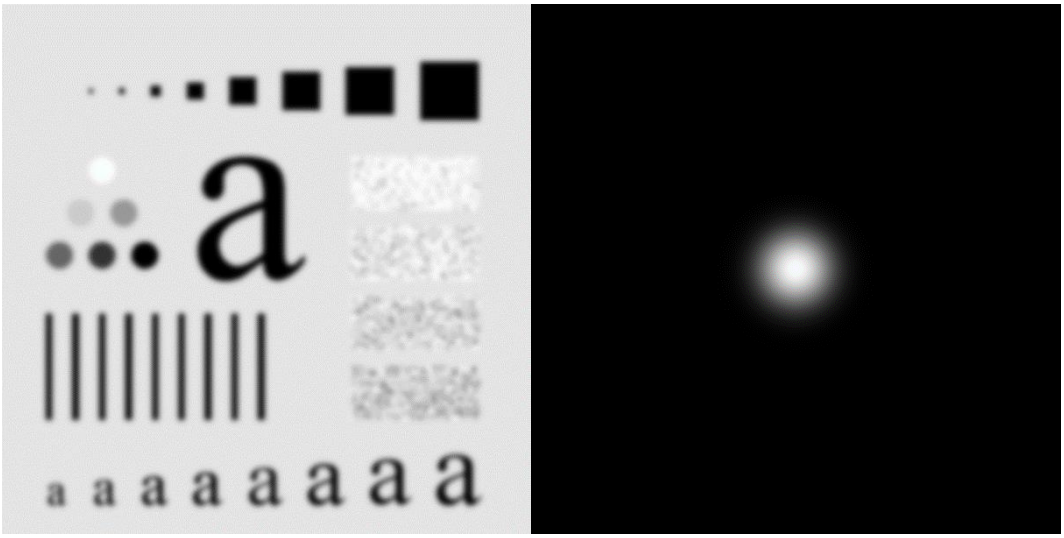


Original Image / Original Frequency Domain

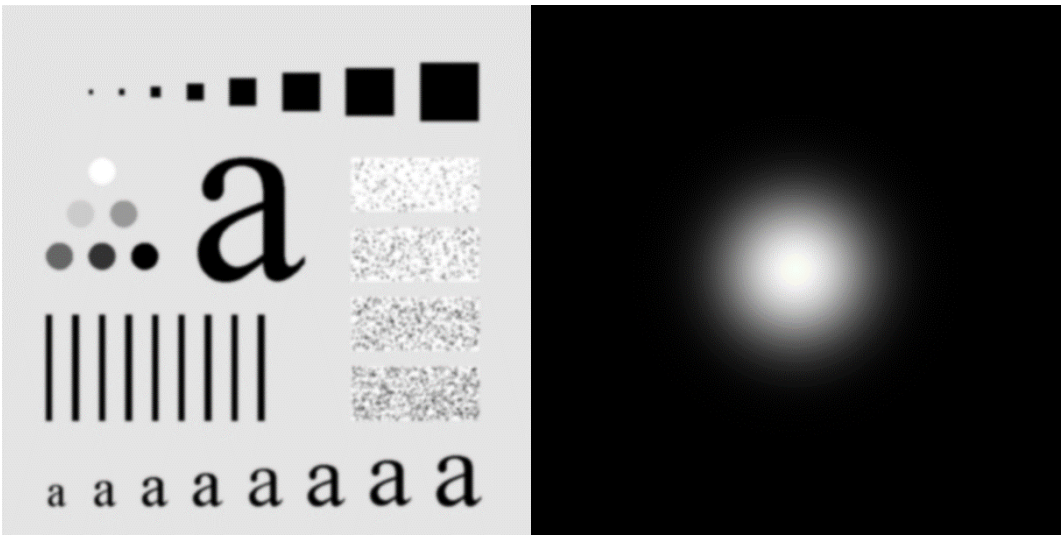


Radii = 10 (Result) / Radii = 10 (Mask)

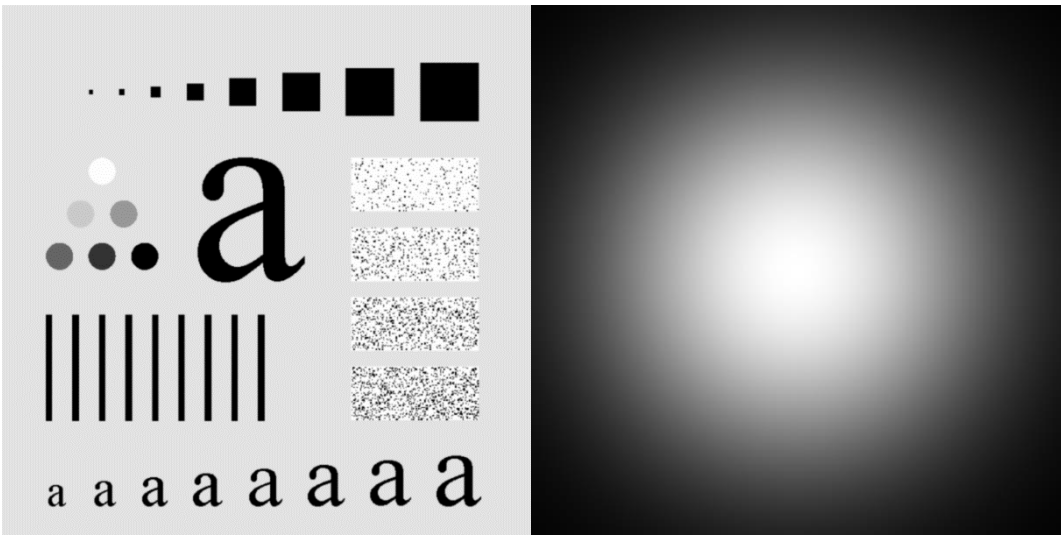
Radii = 30 (Result) / Radii = 30 (Mask)



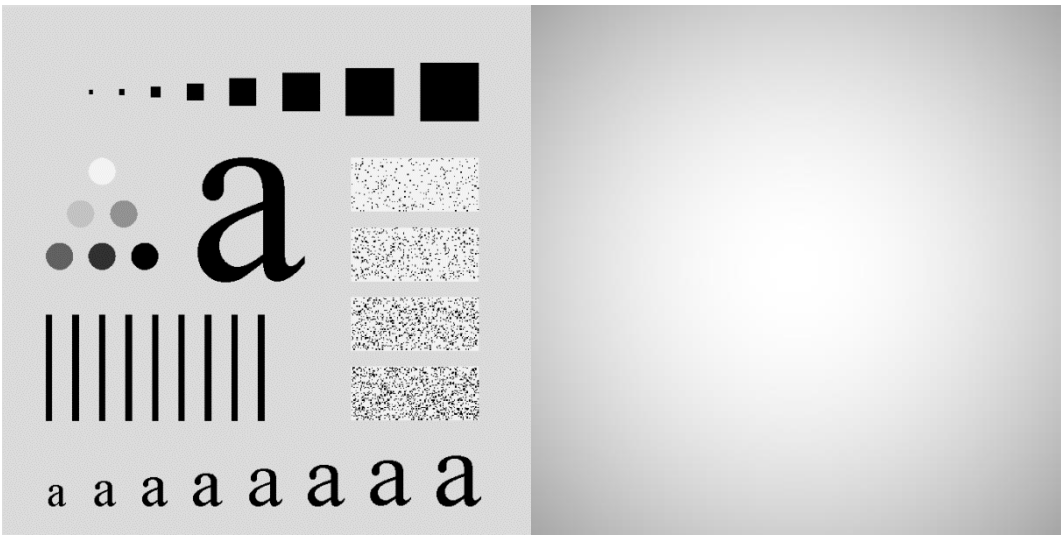
Radii = 60 (Result) / Radii = 60 (Mask)



Radii = 160 (Result) / Radii = 160 (Mask)



Radii = 460 (Result) / Radii = 460 (Mask)



Discussion

可以從結果發現，在 radii 比較小的時候，影像會變得比較模糊，是因為 Lowpass filter 會消除掉頻率較高的部分。當 radii 變高時，影像會變得越來越清楚。

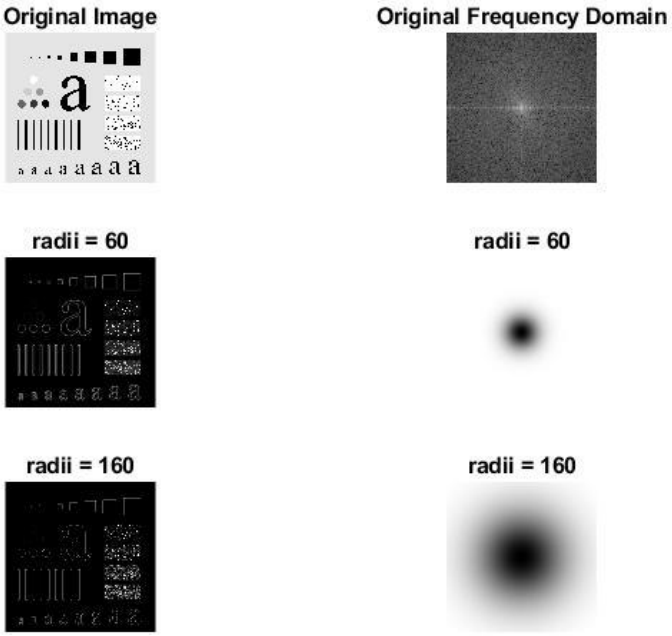
Proj04-04

Explanation

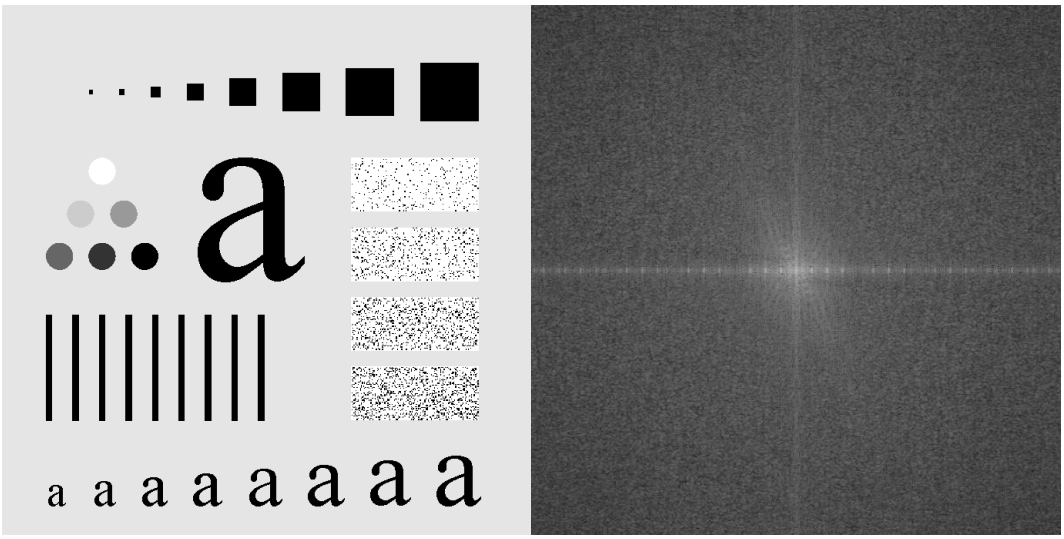
在 myGHPF(D0,M,N)中，我利用 for loop 的方式去時做出大小為 M*N 的 Gaussian highpass filter。基本上是根據上一題的 myGLPF(D0,M,N)去進行修改。

```
H(i,j)= 1 - exp(-((i-M/2)^2+(j-N/2)^2)/(2*D0^2));
```

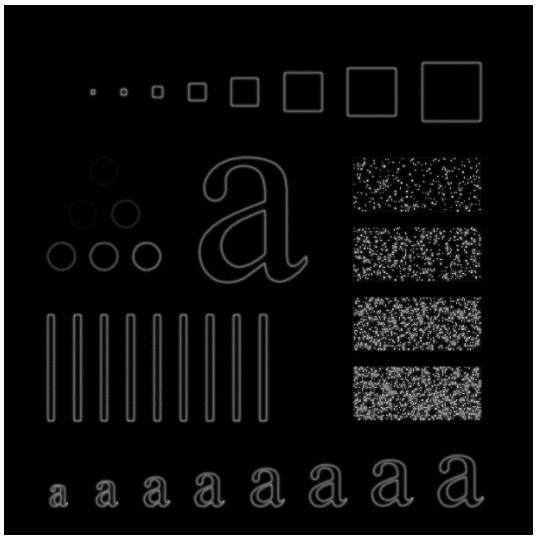
Results

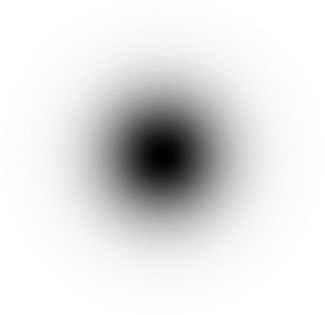


Original Image / Original Frequency Domain

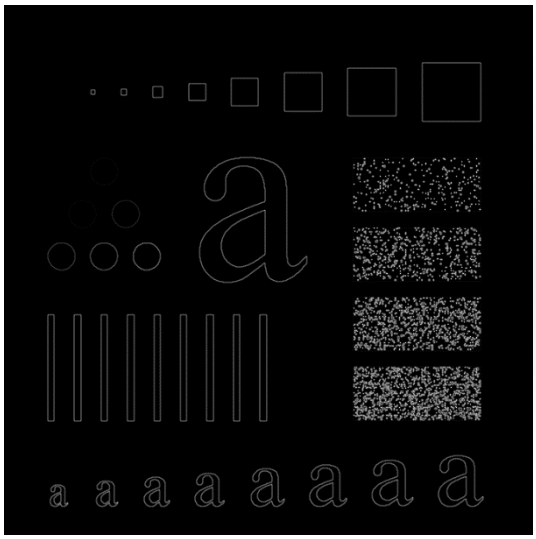


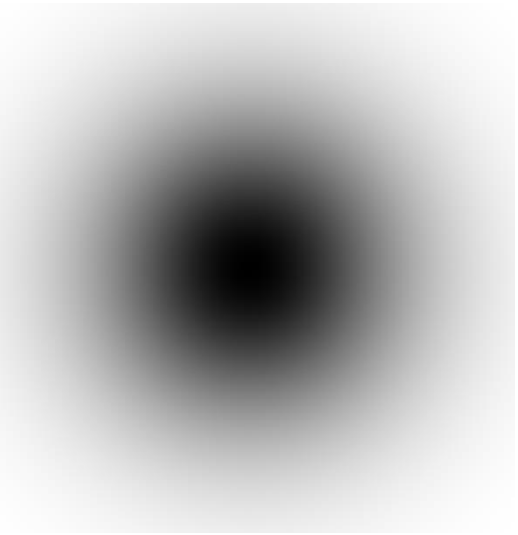
Radii = 60 (Result) / Radii = 60 (Mask)





Radii = 160 (Result) / Radii = 160 (Mask)





Discussion

在 Highpass filter 的實作上，其實是根據前一題的 Lowpass filter 進行修改。接著我利用 Highpass Filter 去實作一張指紋的圖，並且將 D_0 調整成 3，可以發現指紋的紋路有變得稍微明顯一些。但是如果將 D_0 調整成更大，就會使整個影像變得很模糊，因此我認為儘管使用 Highpass filter 最重要的問題是如何去挑選最適合的 D_0。

Radii = 3 (Result) / Radii = 3 (Mask)