

Digital Image Processing

數位影像處理

Homework2

課號 : CO6041
教師 : 唐之瑋
助教 : 詹豐鎧
林邑豪

姓名 : 楊中毅
學號 : 110327007

一、實驗步驟說明

1-1. (先進行一次下取樣) **2-D FFT 運算**

1-2. 2-D IFFT 運算, 並計算轉換回時域圖片與原圖的 **MES, PSNR**

1-3. 加上 **Uniform Noise** 並計算與原圖的 **MES, PSNR**

1-4. 將 1-3. 加入雜訊後影像轉至頻域並進行理想低通濾波

1-5. 計算 1-4. 濾波結果與原圖的 **MES, PSNR**

二、學習目的

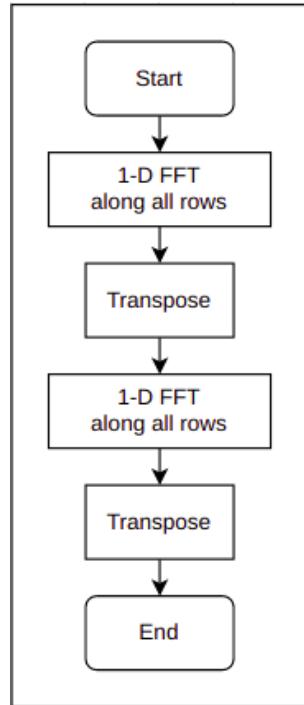
接續實驗一的實驗，了解基本 bmp file 格式，作一些簡單的時域影像處理，下取樣...等。

在實驗二實作 Fast Fourier Transform，將原本時域的影像轉至頻率域處理，並在頻率域作濾波處理。

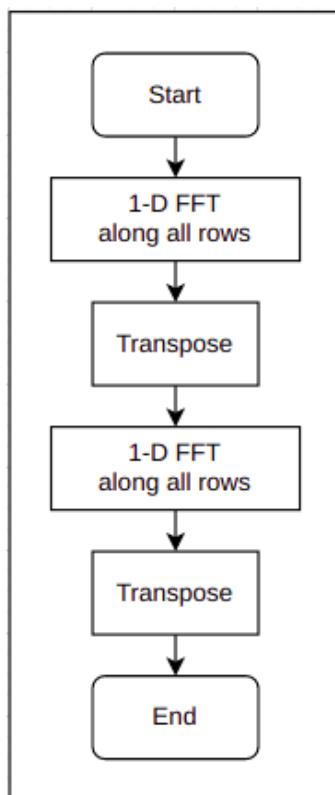
在轉換過程中難免失真或是在某個階段參雜進雜訊，在作業二我們利用 MSE, PSNR 兩個指標來衡量兩個影像的差異性，並以 Uniform Noise 為例，探討此兩參數的意義。

三、實驗步驟流程圖

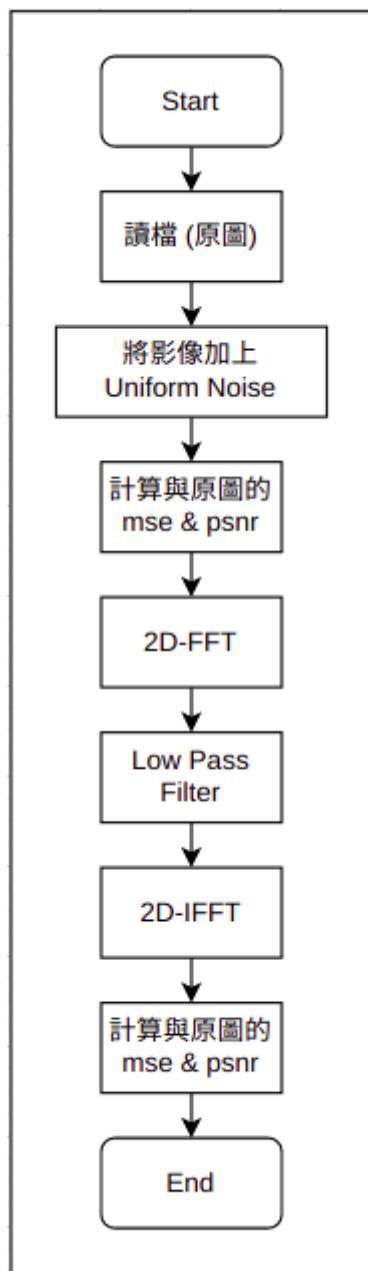
2-D FFT, 2-D IFFT



1-1. 1-2.



1-3. 1-4. 1-5.



四、實驗結果

1-1. (先進行一次下取樣) 2-D FFT 運算

首先進行下取樣，保留偶數 row, 偶數 column。
(詳細實作方法參見 Hw1)

1-D FFT :

以遞迴的方式實作。

另外用一 reserved 空間當作遞迴過程中的運算暫存空間。

```
void fft(int N, Complex_t *data, Complex_t *reserved)
{
    if (N == 1) return ;

    int32_t i;
    Complex_t w, z, *fe, *fo;

    fe = reserved;
    fo = reserved + N / 2;

    for (i = 0; i < N / 2; i++) {
        fe[i] = data[2 * i];
        fo[i] = data[2 * i + 1];
    }

    fft(N/2, fe, data);
    fft(N/2, fo, data);

    for (i = 0; i < N / 2; i++) {
        w.re = cos(2 * PI * (double)i / (double)N);
        w.im = -sin(2 * PI * (double)i / (double)N);
        z.re = w.re * fo[i].re - w.im * fo[i].im;
        z.im = w.re * fo[i].im + w.im * fo[i].re;
        data[i].re = fe[i].re + z.re;
        data[i].im = fe[i].im + z.im;
        data[i + N / 2].re = fe[i].re - z.re;
        data[i + N / 2].im = fe[i].im - z.im;
    }
}
```

2-D FFT :

再開始作 FFT 前，先將原資料乘上 $(-1)^{i+j}$ 作相位平移。
對每個 row 進行 1-D FFT，再將整個影像轉置，對
每個 row 再進行一次 1-D FFT (轉置再對每個 row 作 FFT 等於
對原影像的每個 Column 作 FFT)，最後再轉置回來。

ps. 轉置整個矩陣需要花費時間複雜度 $O(n^2)$ ，應該還可以
找到更好的方式替代。

```
void imgfft2(Image_t *img)
{
    Complex_t *tmprow, *tmpcol;
    uint64_t i, j;

    tmprow = (Complex_t *)calloc(img->width, sizeof(Complex_t));
    tmpcol = (Complex_t *)calloc(img->height, sizeof(Complex_t));

    for (i = 0; i < img->height; i++) {
        for (j = 0; j < img->width; j++) {
            double factor = (i + j) % 2 ? -1 : 1;
            img->freq[i*img->width+j].re = img->data[i*img->width+j] * factor;
        }
    }

    for (i = 0; i < img->height; i++) {
        fft(img->width, (Complex_t *)(&img->freq[img->width*i]), tmprow);
    }
    cTranspose(img->freq, img->height, img->width);
    for (int i = 0; i < img->width; i++) {
        fft(img->height, (Complex_t *)(&img->freq[img->width*i]), tmpcol);
    }
    cTranspose(img->freq, img->height, img->width);

    normfreq(img);

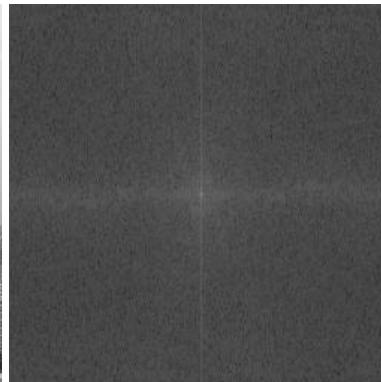
    free(tmprow);
    free(tmpcol);
}
```



(a)



(b)



(c)

Figure 1-1. (a) 原圖 (b) 原圖進行一次下取樣結果 (c) 將圖 (b)
進行 2D-FFT 轉至頻率域。

1-2. 2-D IFFT 運算, 並計算轉換回時域圖片與原圖的 MES, PSNR

1-D IFFT :

與 1-D FFT 幾乎相同, 差在 FFT 中 $e^{-j\theta}$ 項與 IFFT 中 $e^{j\theta}$ 相, 角度正負值不同。

另外本次實作在 FFT 結果乘以係數 1,

在 IFFT 結果成以係數 $\frac{1}{Image\ Size}$.

- The discrete Fourier transform of a function (image) $f(x,y)$ of size $M \times N$

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(ux/M+vy/N)}$$

- The inverse Fourier transform

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(ux/M+vy/N)}$$

```
void ifft(int N, Complex_t *data, Complex_t *reserved)
{
    if (N == 1) return ;

    int32_t i;
    Complex_t w, z, *fe, *fo;

    fe = reserved;
    fo = reserved + N / 2;

    for (i = 0; i < N / 2; i++) {
        fe[i] = data[2 * i];
        fo[i] = data[2 * i + 1];
    }

    ifft(N/2, fe, data);
    ifft(N/2, fo, data);

    for (i = 0; i < N / 2; i++) {
        w.re = cos(2 * PI * (double)i / (double)N);
        w.im = sin(2 * PI * (double)i / (double)N);
        z.re = w.re * fo[i].re - w.im * fo[i].im;
        z.im = w.re * fo[i].im + w.im * fo[i].re;
        data[i].re = fe[i].re + z.re;
        data[i].im = fe[i].im + z.im;
        data[i + N / 2].re = fe[i].re - z.re;
        data[i + N / 2].im = fe[i].im - z.im;
    }
    for (i = 0; i < N; i++) {
        data[i].re /= 2;
        data[i].im /= 2;
    }
}
```

2-D IFFT：

與 2-D FFT 幾乎相同。

先不作相位平移，依序做完 IFFT 後，將每筆資料乘上 $(-1)^{i+j}$ ，還原 2-D IFFT 前作的相位平移。

```
void imgIFFT2(Image_t *img)
{
    Complex_t *tmprow, *tmpcol;
    uint64_t i, j;

    tmprow = (Complex_t *)calloc(img->width, sizeof(Complex_t));
    tmpcol = (Complex_t *)calloc(img->height, sizeof(Complex_t));

    for (int i = 0; i < img->width; i++) {
        ifft(img->height, (Complex_t *)(&img->freq[i*img->width*i]), tmpcol);
    }
    cTranspose(img->freq, img->height, img->width);
    for (i = 0; i < img->height; i++) {
        ifft(img->width, (Complex_t *)(&img->freq[i*img->width*i]), tmprow);
    }
    cTranspose(img->freq, img->height, img->width);

    for (i = 0; i < img->height; i++) {
        for (j = 0; j < img->width; j++) {
            double factor = (i + j) % 2 ? -1 : 1;
            img->data[i*img->width+j] = img->freq[i*img->width+j].re * factor;
        }
    }
    free(tmprow);
    free(tmpcol);
}
```

做完 FFT 與 IFFT 後，將結果與原圖比較，本次作業使用 PSNR 與 MSE 這兩個指標。

計算函式如下：

```
double mse(uint8_t *img1, uint8_t *img2, uint64_t size)
{
    double err = 0;
    for (int i = 0; i < size; i++) {
        err += (img1[i] - img2[i]) * (img1[i] - img2[i]);
    }
    return err / size;
}

double psnr(uint8_t *img1, uint8_t *img2, uint64_t size)
{
    return 10 * log10(255*255 / mse(img1, img2, size));
}
```

HW 2-2 所求

MSE = 0.188675

PSNR = 55.373662 dB

可以觀察到 MSE 極小，PSNR 夠大，代表我們將原圖轉至頻域再轉回時域的失真極小。理論上傅立葉轉換可以無失真轉回原函式，但考慮到電腦系統浮點數的儲存與計算的有限性，在本次實驗中產生的誤差還不算太大，可以接受。

```
cy023@cy023-e14:~/project/DIP-Fall2021/FFT$ ./hw2.out
[HW2-1] : Downsample the Image and 2D-FFT.

[HW2-2] : After 2D-FFT and 2D-IDFT.
    mse = 0.188675
    psnr = 55.373662 dB

[HW2-3] : After Adding Uniform Noise to the Image.
    mse = 139.527901
    psnr = 26.684193 dB

[HW2-4] : Ideal Low Pass Filter.

[HW2-5] : After Processing by Ideal Low Pass Filter.
    mse = 285.207027
    psnr = 23.579201 dB
```



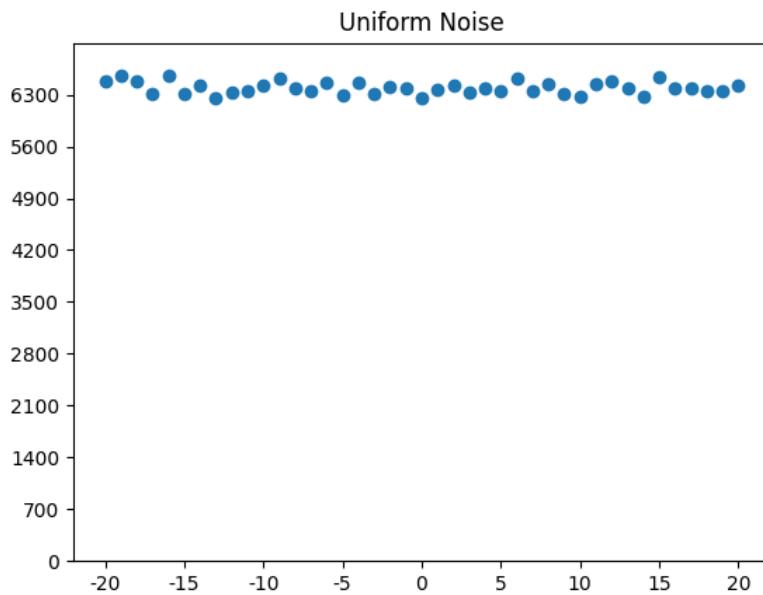
Figure 1-2. 將 Figure 1-1 (c) 影像轉回時域結果圖。

1-3. 加上 Uniform Noise 並計算與原圖的 MSE, PSNR

Uniform Noise :

```
void uniform_noise(int16_t *noise, uint64_t nsize, int16_t a, int16_t b)
{
    srand(time(NULL));
    uint64_t i, j;
    int16_t min = a < b ? a : b;
    int16_t range = abs(b - a);
    for (i = 0; i < nsize; i++) {
        noise[i] = (rand() % (range + 1)) + min;
    }
}
```

將隨機產生的 Uniform Noise 印出，利用 python 將資料可視化，可以看到雜訊的確平均分佈。



HW 2-3 所求

MSE = 139.527901

PSNR = 26.684193 dB

加上 (-20,20) 的 Uniform 雜訊後，MSE 指標相較 HW2-2 的 MSE 還大，PSNR 指標相較 HW2-2 的 PSNR 還小。

不管在主觀的感受看起來多了很多雜點，或是客觀的 MSE, PSNR 統計指標上都能看出影像品質下降。

```
cy023@cy023-e14:~/project/DIP-Fall2021/FFT$ ./hw2.out

[HW2-1] : Downsample the Image and 2D-FFT.

[HW2-2] : After 2D-FFT and 2D-IFFT.
    mse = 0.188675
    psnr = 55.373662 dB

[HW2-3] : After Adding Uniform Noise to the Image.
    mse = 139.527901
    psnr = 26.684193 dB

[HW2-4] : Ideal Low Pass Filter.

[HW2-5] : After Processing by Ideal Low Pass Filter.
    mse = 285.207027
    psnr = 23.579201 dB
```

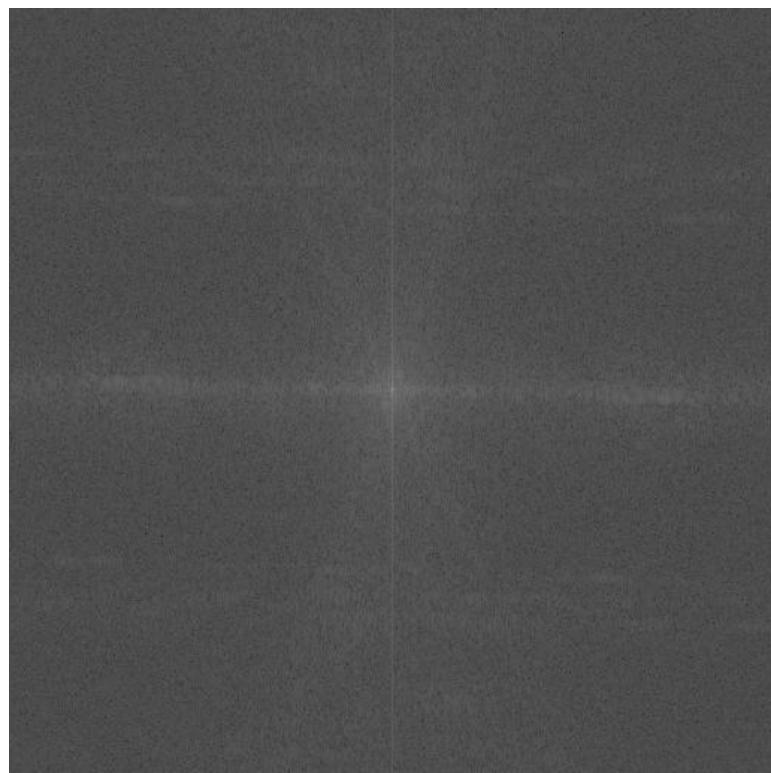


Figure 1-3. 將原圖加上 (-20,20) 的 Uniform Noise.

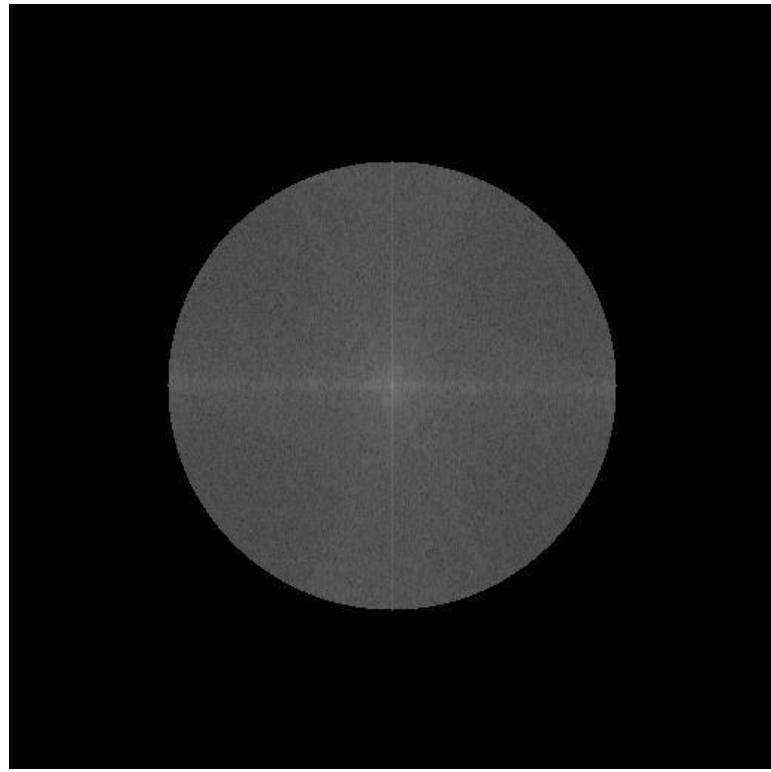
1-4. 將 1-3. 加入雜訊後影像轉至頻域並進行理想低通濾波

將加入雜訊的影像轉至頻域。
並通過理想低通濾波。

```
void lowpass(Image_t *img, int16_t cutoff)
{
    int16_t i, j;
    int16_t M = img->height;
    int16_t N = img->width;
    for (i = 0; i < img->height; i++) {
        for (j = 0; j < img->width; j++) {
            if (sqrt(pow(i-M/2, 2) + pow(j-N/2, 2)) > cutoff) {
                img->freq[i * img->width + j].re = 0;
                img->freq[i * img->width + j].im = 0;
            }
        }
    }
    normfreq(img);
}
```



(a)



(b)

Figure 1-4. (a) 將 Figure 1-3. 進行 2D-FFT 運算轉換至頻率域。
(b) 將 Figure 1-4.(a) 頻域資料進行理想低通濾波 ($D_0 = 150$).

1-5. 計算 1-4. 濾波結果與原圖的 MSE, PSNR

將通過理想低通濾波的的頻域資料轉回時域。

HW 2-5 所求

MSE = 285.207027

PSNR = 23.579201 dB

加入雜訊再經過低通濾波器濾除高頻訊號後，影像失真很多，比只加入雜訊還糟。

```
cy023@cy023-e14:~/project/DIP-Fall2021/FFT$ ./hw2.out
[HW2-1] : Downsample the Image and 2D-FFT.

[HW2-2] : After 2D-FFT and 2D-IDFT.
    mse = 0.188675
    psnr = 55.373662 dB

[HW2-3] : After Adding Uniform Noise to the Image.
    mse = 139.527901
    psnr = 26.684193 dB

[HW2-4] : Ideal Low Pass Filter.

[HW2-5] : After Processing by Ideal Low Pass Filter.
    mse = 285.207027
    psnr = 23.579201 dB
```



Figure 1-5. 將 Figure 1-4.(b) 影像轉回時域結果圖。

五、分析

影像品質指標

MES (Mean Squared Error)

$$MSE = \frac{1}{Image\ Size} \sum_{i=1}^{Image\ Size} (y_i - \hat{y}_i)^2$$

y_i 為原圖數值 , \hat{y}_i 為要比較影像的數值 , 由上述公式能知道當兩張圖片 MSE 越大, 失真越大。

PSNR (Peak Signal-to-Noise Ratio)

$$PSNR = 10 \times \log \left(\frac{P^2}{MSE} \right)$$

PSNR 在計算最大能量與雜訊能量的比值，以 dB 為單位。數值越大代表影像品質越好。

在 hw2 中, 以 8 bit 灰階圖像進行實驗, 公式中最大能量 P 以 255 帶入計算。(補充:若是 HDR 影像, 以各位元深度分別計算)

維基百科中對於 psnr 值的意義說明：

圖像與影像壓縮中典型的峰值訊噪比值在 30dB 到 50dB 之間，愈高愈好。

- PSNR 接近 50dB，代表壓縮後的圖像僅些許非常小的誤差。
- PSNR 大於 30dB，人眼很難察覺壓縮後和原始影像的差異。
- PSNR 介於 20dB ~ 30dB，人眼就可以察覺出圖像的差異。
- PSNR 介於 10dB ~ 20dB，人眼還是可以用肉眼看出這個圖像原始的結構，且直觀上會判斷兩張圖像不存在很大的差異。
- PSNR 低於 10dB，人類很難用肉眼去判斷兩個圖像是否為相同，一個圖像是否為另一個圖像的壓縮結果。

SSIM (Structure Similarity)

因為 PSNR 與 MSE 缺少空間特性，僅計算每個像素的差值，不能完全貼切影像品質的主觀判斷。

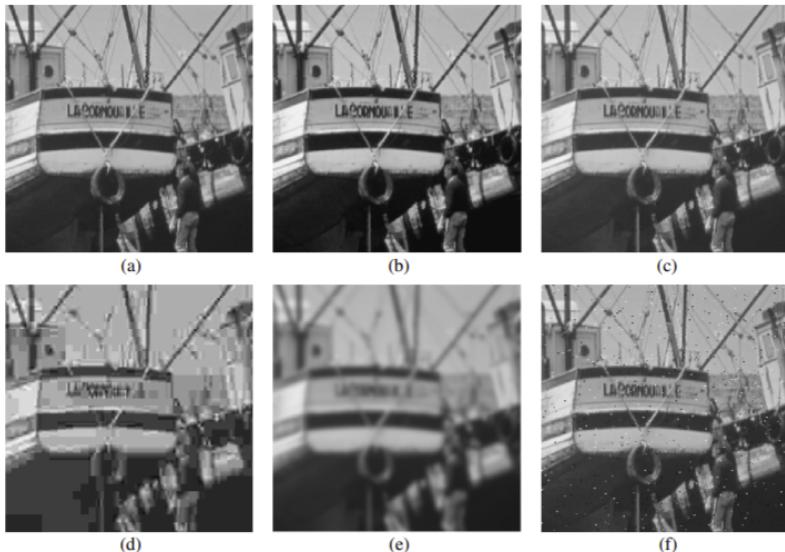


Fig. 2. Comparison of "Boat" images with different types of distortions, all with $MSE = 210$. (a) Original image (8 bits/pixel; cropped from 512×512 to 256×256 for visibility). (b) Contrast-stretched image, $MSSIM = 0.9168$. (c) Mean-shifted image, $MSSIM = 0.9900$. (d) JPEG compressed image, $MSSIM = 0.6949$. (e) Blurred image, $MSSIM = 0.7052$. (f) Salt-pepper impulsive noise contaminated image, $MSSIM = 0.7748$.

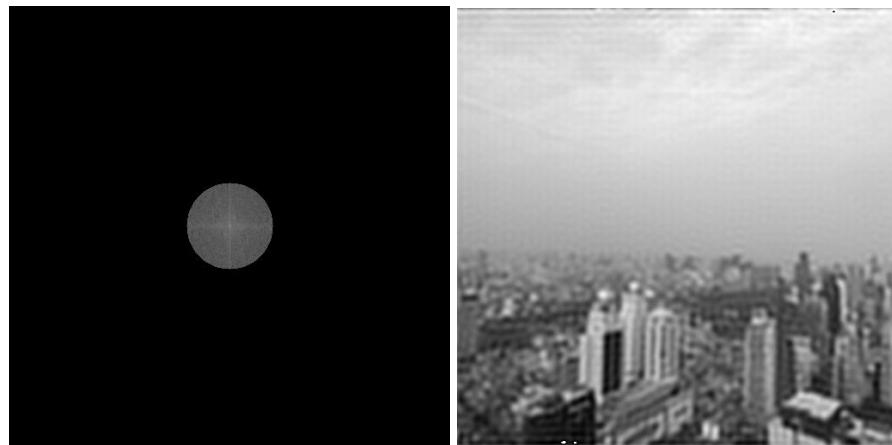
為了更符合 HVS (人類視覺系統) 提出 SSIM 指標。詳細請見 Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.

理想低通濾波器比較

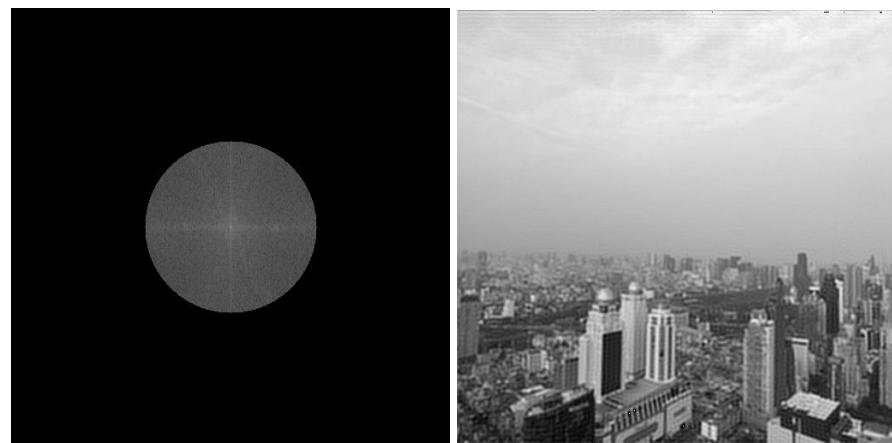
由結果可以看出低通濾波器會將高頻成份濾除，當截止頻率越小，保留下來的影像都是低頻成份，整體影像模糊。

低通濾波截止頻率越高，失真越小 (MSE 小, PSNR 大)

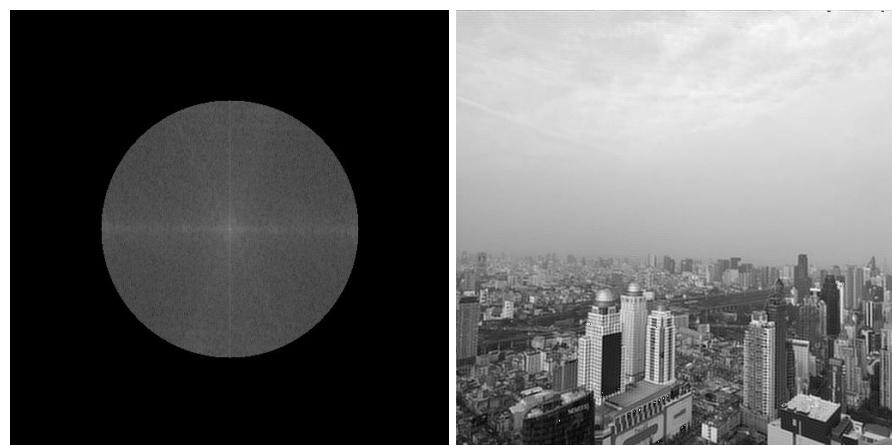
$D_0 = 50$ (MSE = 444.302292, PSNR = 21.654018 dB)



$D_0 = 100$ (MSE = 325.229671, PSNR = 23.008902 dB)



$D_0 = 150$ (MSE = 239.795990, PSNR = 24.332384 dB)

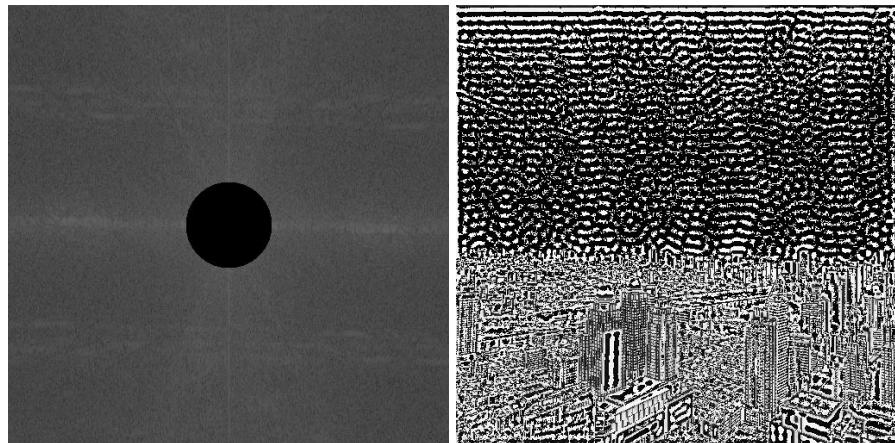


理想高通濾波器比較

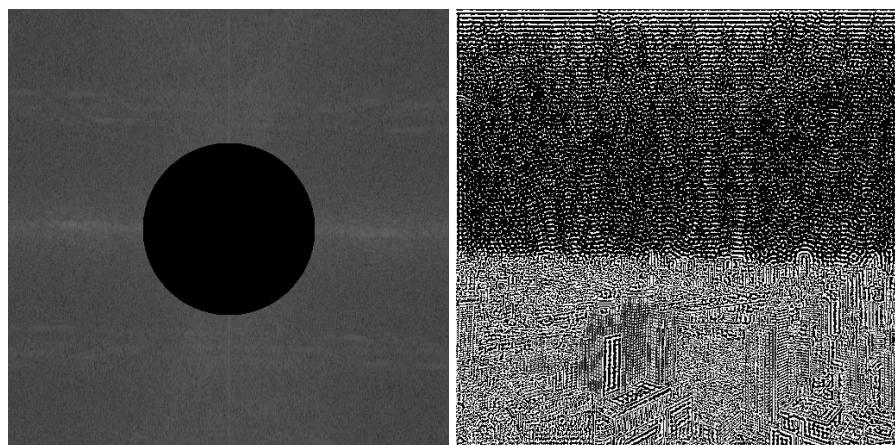
高通濾波器會將低頻成份濾除，保留影像中高頻成份。

高通濾波截止頻率越低，失真越小 (MSE 小, PSNR 大)

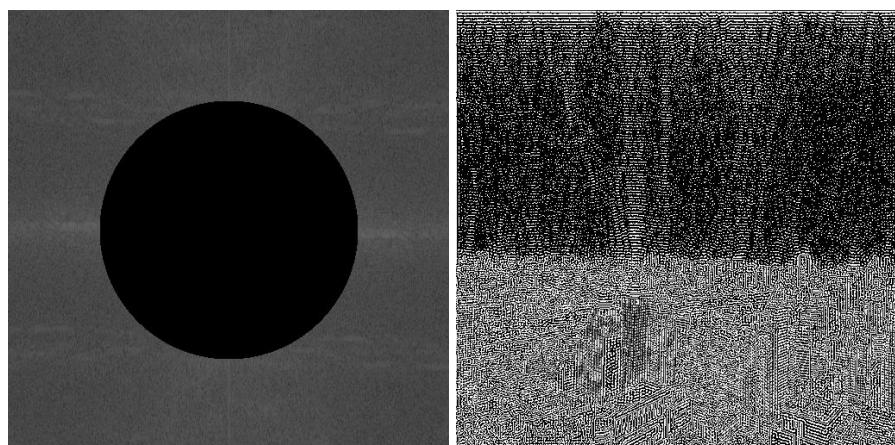
$D_0 = 50$ (MSE = 26570.563908, PSNR = 3.886796 dB)



$D_0 = 100$ (MSE = 28236.180965, PSNR = 3.622744 dB)



$D_0 = 150$ (MSE = 28624.859203, PSNR = 3.563370 dB)



Uniform Noise 比較

雜訊波動越大，圖片看起來越糟，失真越多 (MSE 大, PSNR 小)

$a = -10, b = 10$

$\text{mse} = 36.591106$

$\text{psnr} = 32.497048 \text{ dB}$



$a = -20, b = 20$

$\text{mse} = 139.687683$

$\text{psnr} = 26.679222 \text{ dB}$



$a = -30, b = 30$

$\text{mse} = 304.854851$

$\text{psnr} = 23.289873 \text{ dB}$



六、Reference

- [1] Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.
- [2]<https://zh.wikipedia.org/wiki/%E5%B3%B0%E5%80%BC%E4%BF%A1%E5%99%AA%E6%AF%94>