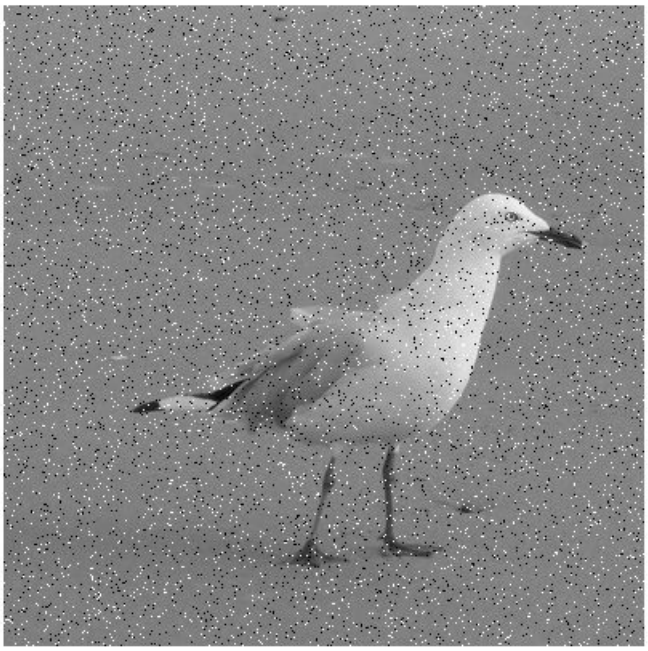
# 影像修復 (Imager Restoration)

雜訊有分成很多種，下面展示不同種類的雜訊

## Salt and Pepper Noise (Impulse Noise, Shot Noise, Binary Noise)

這種雜訊主要就是在影像上隨機撒上黑點跟白點

```
img = rgb2gray(imread("Image\gull.png"));
gsp = imnoise(img, "salt & pepper");
figure; imshow(gsp);
```



## Gaussian Noise

這種雜訊就是原圖加上一個常態分配的訊號圖，可被定義為

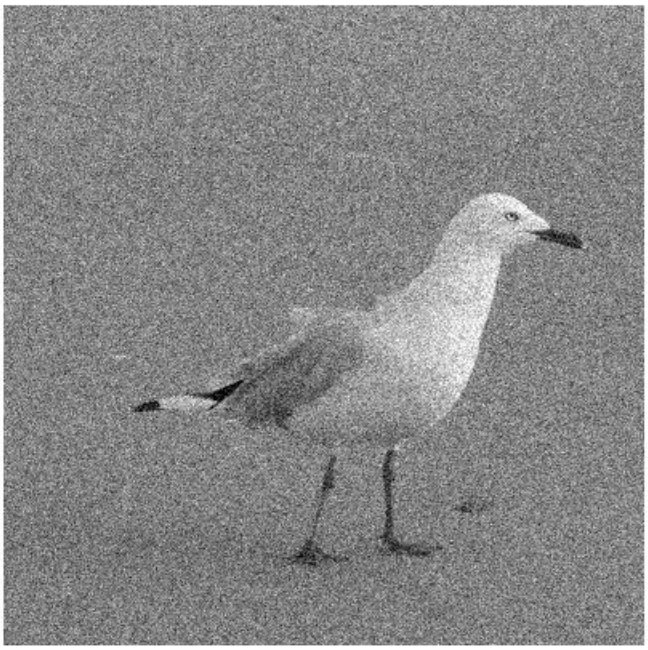
$K = I + N$

其中 $K$: 有雜訊的影像

$I$ : 原圖

$N$ : 平均數為 0 的 normal distribution

```
gg = imnoise(img, "gaussian");
figure; imshow(gg);
```



## Speckle Noise (Multilicative Noise)

這種雜訊就是原圖乘上一個雜訊圖，可被定義為

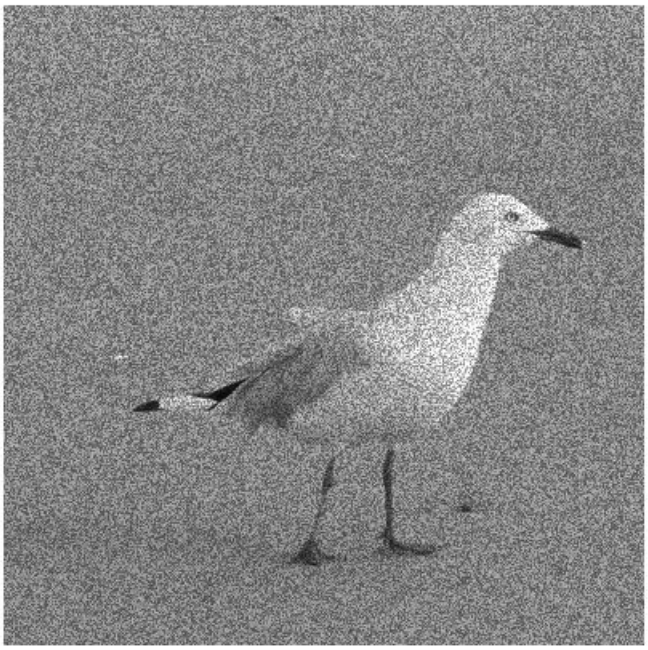
$K = I(1 + N)$

其中 $K$: 有雜訊的影像

$I$ : 原圖

$N$ : 平均數為 0 的 normal distribution

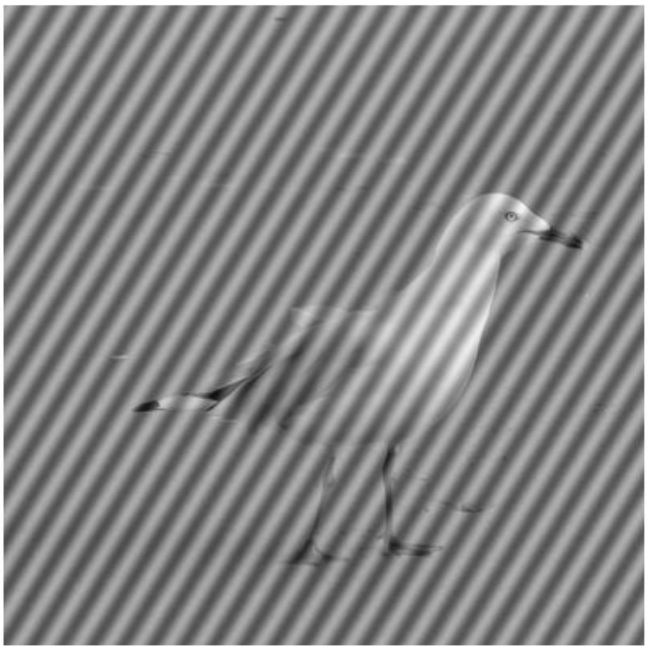
```
gs = imnoise(img, "speckle");
figure; imshow(gs);
```



## Periodic Noise

就是帶有週期性的雜訊

```
[r, c] = size(img);
[x, y] = meshgrid(1:r, 1:c);
p = sin(x/3 + y/5) + 1;
gp = (2*im2double(img) + p/2)/3;
figure; imshow(gp);
```



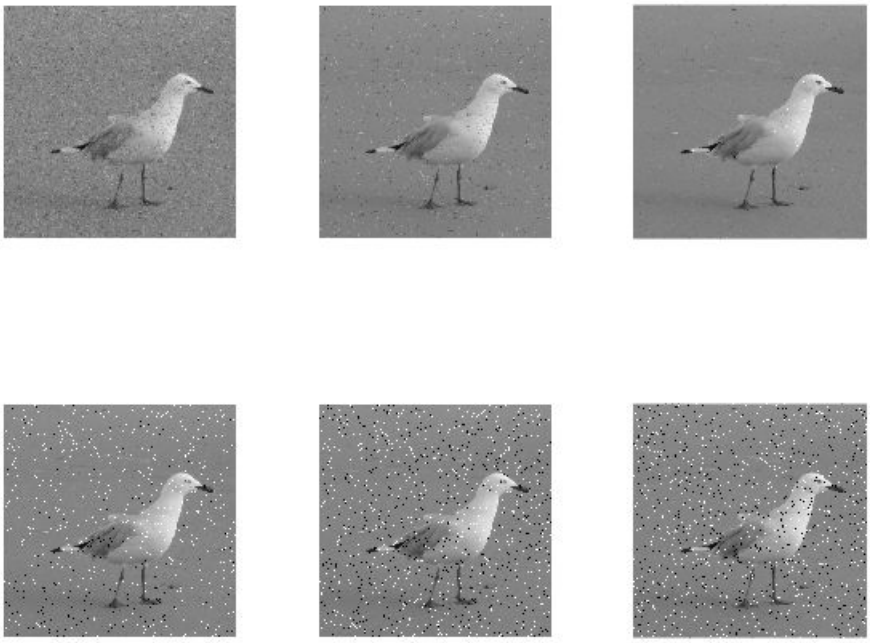
## Salt and Pepper Noise Restoration

要 filter out salt and pepper noise 可以使用 low pass filter 或者是 median filter，但使用 low pass filter 就要承受影像變模糊的副作用。所以使用 median filter 會是比較好的方法。

```
f = fspecial("average", 3);
lp = imfilter(gsp, f);
md = medfilt2(gsp, [3, 3]);
figure;
subplot(121); imshow(lp);
subplot(122); imshow(md);
```

可以看到 median filter 漂亮的移除了 noise，而且還跟原圖差不多。另外我們也可以使用 outlier filter，給定一個 threshold D，當一個 pixel 減掉該 neighborhood 的值大於 D 時，我們便認定他為 outlier，將其設成 neighborhood 的平均

```
gsp2 = im2double(gsp);
av = [1 1 1; 1 0 1; 1 1 1] / 8;
gspa = imfilter(gsp2, av);
D = [0.05 0.1 0.3 0.5 0.7 0.9];
figure;
for i = (1:6)
    d = D(i);
    r = abs(gsp2 - gspa) > d;
    subplot(2, 3, i);
    imshow(r.*gspa + (1-r).*gsp2);
end
```

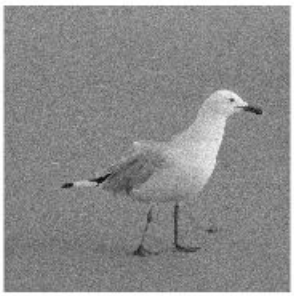
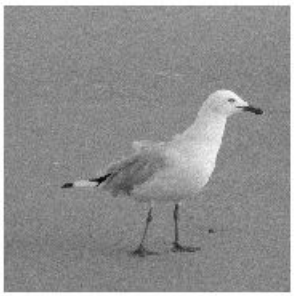
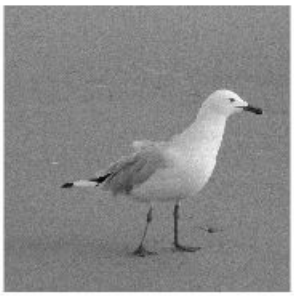
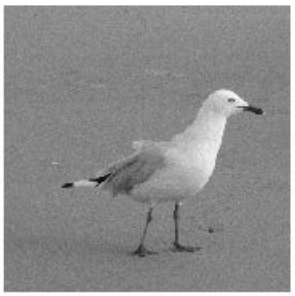


可以看到 threshold 的挑選是很重要的，threshold 太小便會誤把正常的 pixel 當作 outlier，使的影像變模糊。Threshold 太大便會偵測不到 outlier。

## Gaussian Noise Restoration

因為 Gaussian Noise 是影像加上一個平均為 0 的 normal distribution 組成，所以其中一種 filter out 的方法就是把很多張帶有 Gaussian Noise 的影像做平均

```
rep = [5 10 15 20];
figure;
for i = (1:4)
    t = zeros([size(gg), rep(i)]);
    for j = 1:rep(i)
        t(:,:,j) = imnoise(img, "gaussian");
    end
    ta = mean(t, 3);
    subplot(2, 2, i); imshow(mat2gray(ta));
end
```

可以看到隨著圖疊的張數越來越多、圖就會越來越乾淨。

## Adaptive Filtering

這種 filter 可以用來 filter out Gaussian noise 同時保留邊緣。概念是要在偵測到圖片邊緣時要使用 median filter, 在 low freq 的地方要使用 average filter，因此單一個 pixel 經過 mask operation 後的 output 值為

$$m_f + \frac{\sigma_f^2}{\sigma_f^2 + \sigma_g^2}(g - m_f)$$

其中 $m_f$ : mask 下的 pixel 平均

$\sigma_f^2$ : mask 下的 pixel 變異數

$\sigma_g^2$ : 整張 image 的 noise 的變異數

$g$ : pixel 值

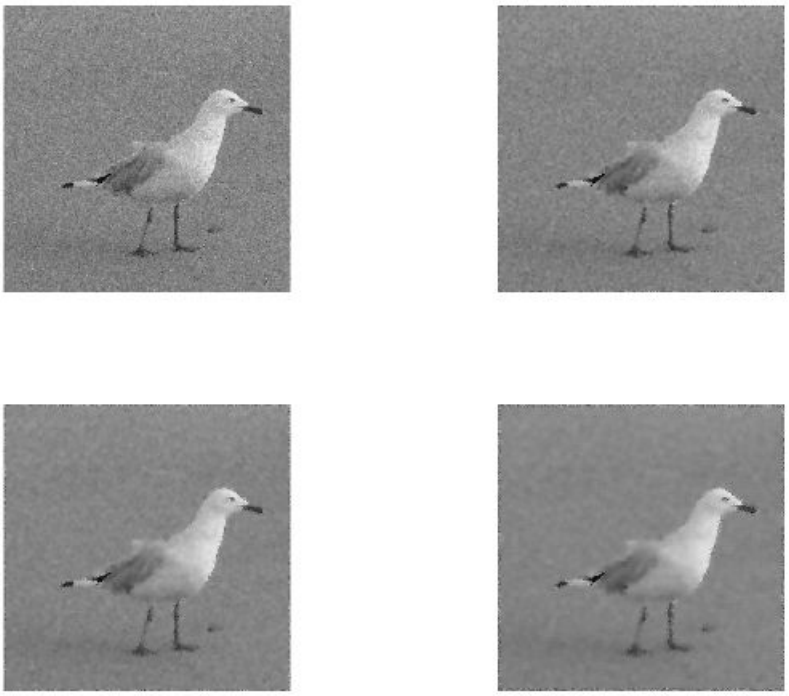
所以這裡可以發現當 $\sigma_f^2$ 越大，代表這個地方是邊緣，輸出的結果就會越接近 $g$。相反的如果 $\sigma_f^2$ 越小，代表這個地方是 low freq，輸出結果就會越接近 $m_f$。

但實際上 $\sigma_g^2$ 不會知道，所以上面的公式會用下面的這個來替代，稱作 Wiener Filter

$$m_f + \frac{\max\{0, \sigma_f^2 - n\}}{\max\{\sigma_f^2, n\}}(g - m_f)$$

其中 $n$ : 所有 $\sigma_f^2$ 的平均。

```
n = [3 5 7 9];
figure;
for i = (1:4)
    wimg = wiener2(gg, [n(i), n(i)]);
    subplot(2, 2, i); imshow(wimg);
end
```



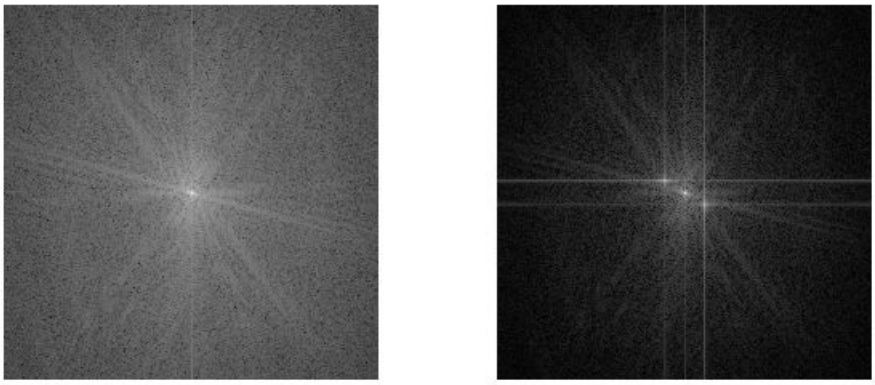
可以看到 Wiener Filter 的 size 越大，濾的效果越好，而且他同時也能保留邊緣的部分。

## Periodic Noise Restoration

這邊我們觀察原圖的 Spectrum 跟有 Periodic Noise 的 Spectrum，再移掉新的 Spectrum 中多的部分再用 iFFT 轉回來。但要注意的點是這種方法需要先有原圖才能實現

```
gf = fftshift(fft2(img));
gpf = fftshift(fft2(gp));
figure;
subplot(121); imshow(mat2gray(log(1 + abs(gf))))
```

```
subplot(122); imshow(mat2gray(log(1 + abs(gpf))));
```



左圖是原圖的 Spectrum，右圖是有 periodic noise 的 spectrum。可以看到右圖比左圖多了兩個亮點以及四條線。接下來我們嘗試把那兩個亮點移除。

```
gpf2 = im2uint8(mat2gray(abs(gpf)));
gpf2(201, 201) = 0; % 先把中心的 DC 設為 0
[i, j] = find(gpf2 == max(gpf2(:))) % 找到那兩個亮點
```

```
i = 2×1
   188
   214
j = 2×1
   180
   222
```

```
(i-201).^2 + (j-201).^2 % 發現兩個點距離中心的距離平方皆為 610
```

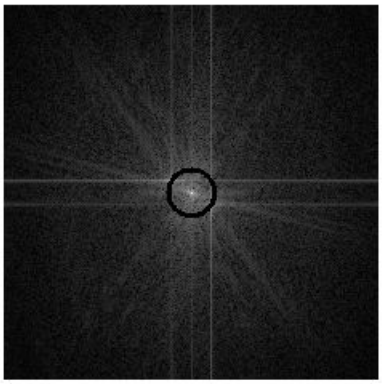
```
ans = 2×1
   610
   610
```

```
rd = sqrt(610)
```

```
rd = 24.6982
```

```
k = 2; % 帶寬
[r, c] = size(gpf2);
[x, y] = meshgrid(1:r, 1:c);
z = sqrt((x-201).^2 + (y-201).^2);
mask = (z < floor(rd - k) | z > ceil(rd + k)); % 畫出遮罩
```

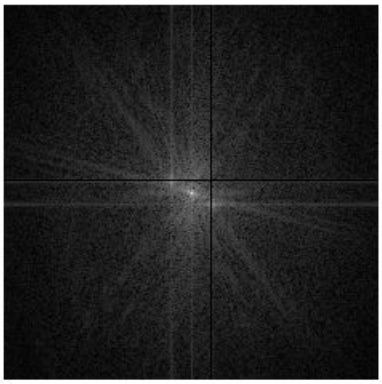
```
gpf3 = gpf.*mask;
new_img = ifft2(gpf3);
figure;
subplot(121); imshow(mat2gray(log(1 + abs(gpf3))));
subplot(122); imshow(abs(new_img));
```



左圖為遮罩遮掉的部分，右圖為轉換後的結果，可以看到中間部分的 periodic noise 被過濾掉了。接著用 iFFT 將 Spectrum 轉成影像後就成功過濾掉了部分的 periodic noise，這種作法稱作 ***Band Reject Filtering*** 或 ***Notch FIltering。***

另外一種作法也可以 block 掉四條直線

```
gf_line = gpf;
gf_line(i, :) = 0;
gf_line(:, j) = 0;
new_img_line = ifft2(gf_line);
figure;
subplot(121); imshow(mat2gray(log(1 + abs(gf_line))));
subplot(122); imshow(abs(new_img_line));
```



這種作法稱作 ***Criss-Cross Filtering***

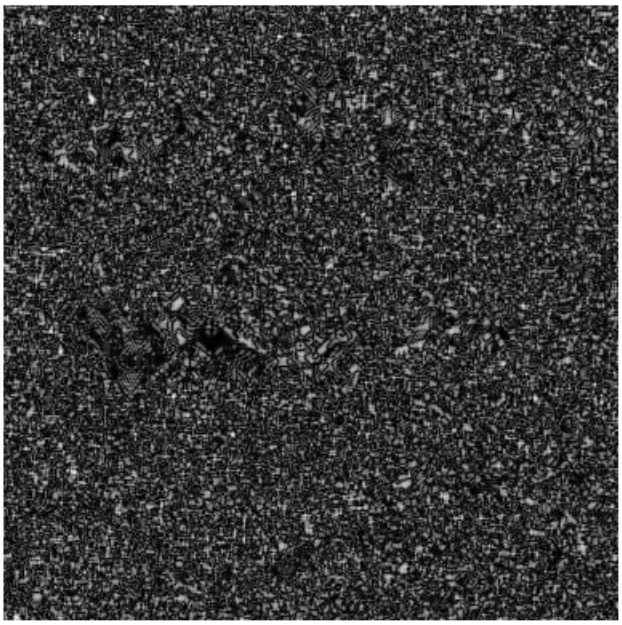
## 反轉濾波 (Inverse)

要將影像在 Spectrum 裡做轉換就是把 spectrum 乘上 spatial filter，也就是說如果要得到轉換前的影像就是把轉換後的影像除以 spatial filter。下面實作反轉模糊影像，首先第一步先用 Butterworth Filter 產生一個模糊影像

```
b = imread("Image\buffalo.png");
bf = fftshift(fft2(b));
[r, c] = size(b);
[x, y] = meshgrid(-c/2:c/2 - 1, -r/2:r/2 - 1);
bworth = 1 ./ (1 + (sqrt(2) - 1) * ((x.^2 + y.^2) / 15^2).^2);
bw = bf.*bworth;
ibw = abs(ifft2(bw)); % ifft 完的影像
blur = im2uint8(mat2gray(ibw)); % 調成影像格式
figure; imshow(blur);
```

接著嘗試將濾波完的影像除以原本的 Butterworth Filter

```
blurft = fftshift(fft2(blur));
blur_inverse = blurft./bworth;
bwa2 = ifft2(blur_inverse);
figure; imshow(mat2gray(abs(bwa2)));
```



可以看到直接變成雜訊，這是因為 Butterworth FIlter 中有一些值很小，會導致除完的值變得過大。要改善有兩種方法，第一種是再乘以一個 Low Pass FIlter 來過濾掉過小的值，這裡我們再乘以一個 Butterworth FIlter

```
bworth2 = 1 ./ (1 + (sqrt(2) - 1) * ((x.^2 + y.^2) / 45^2).^2);
blur_inverse2 = blurft./bworth.*bworth2;
bwa3 = abs(ifft2(blur_inverse2));
figure; imshow(im2uint8(mat2gray(bwa3)));
```

可以看到牛變得比較清楚一點了。第二種方法就是設一個 threshold，當 Butterworth FIlter 小於 threshold 的時候就不要除。

```
d = 0.01;
new_bw = bworth;
new_bw((new_bw < d)) = 1;
blur_inverse3 = blurft./new_bw;
bwa4 = abs(ifft2(blur_inverse3));
figure; imshow(im2uint8(mat2gray(bwa4)));
```

如此便成功反轉影像了，這種方法稱作 ***Constrained Division***