

邊緣銳化 (Edge Sharpening or Edge Enhancement or Unsharp Masking)

為了要讓圖片的邊緣凸顯，然後低頻的部分保持原樣，可以用這種方法。概念上就是將原圖減掉一張經過 low pass filter 的圖，這樣就可以把高頻的部分凸顯出來。其濾波器可以如下表示

$$u = s(id - \frac{1}{k}a)$$

其中 id 為 identity matrix, 也就是原圖的部分

a 為 low pass filter, 這裡假設他是 average filter

k, s 為縮放常數

為了讓新的圖跟原圖有相同的亮度, 設 u 的 element 總合為 1。因此可以導出

$$s(1 - \frac{1}{k}) = 1$$

令 $k = 1.5$, 則 $s = 3$, 所以這個 filter 為

$$3 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - 2 \left(\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) = \frac{1}{9} \begin{bmatrix} -2 & -2 & -2 \\ -2 & 25 & -2 \\ -2 & -2 & -2 \end{bmatrix}$$

```
img = imread("Image\koala.png");  
id = [0 0 0; 0 1 0; 0 0 0];  
a = fspecial("average", [3, 3]);  
u = 3*id - 2*a;  
img2 = imfilter(img, u);  
figure, imshow(img);
```



```
figure, imshow(img2);
```



可以看到在經過 unsharpening mask 之後，細節變的更加的明顯。而下面可以實驗調整 s 讓 u 的 element 總和不是 1 時會發生甚麼事

```
count = 1;
```

```
figure;
for i = [5, 4, 2.5, 2.2]
    u = i*id - 2*a;
    imgf = imfilter(img, u);
    subplot(2, 2, count);
    imshow(imgf);
    count = count + 1;
end
```



可以看到當 s 越大時，圖片越亮，反之則越暗。因為 s 代表著原圖的縮放倍率，當原圖乘的倍數越高時，圖片就會越亮。

fspecial 裡也有內建的 unsharpening mask 可以使用，他是用 Laplacial Filter (High Pass Filter) 來轉換，所以他的運作方式會變成原圖去加上一個經過 High Pass Filter 之後的影像。而其使用的 Filter 為

$$\frac{1}{\alpha + 1} \begin{bmatrix} -\alpha & \alpha - 1 & -\alpha \\ \alpha - 1 & \alpha + 5 & \alpha - 1 \\ -\alpha & \alpha - 1 & -\alpha \end{bmatrix}$$

其中 $0 < \alpha < 1$

```
count = 1;
```

```
figure;
for i = [0.2, 0.4, 0.6, 0.8]
    f = fspecial("unsharp", i)
    imgf = imfilter(img, f);
    subplot(2, 2, count);
    imshow(imgf);
    count = count + 1;
end
```

```
f = 3x3
    -0.1667    -0.6667    -0.1667
    -0.6667     4.3333    -0.6667
    -0.1667    -0.6667    -0.1667
```

```
f = 3x3
    -0.2857    -0.4286    -0.2857
    -0.4286     3.8571    -0.4286
    -0.2857    -0.4286    -0.2857
```

```
f = 3x3
    -0.3750    -0.2500    -0.3750
    -0.2500     3.5000    -0.2500
    -0.3750    -0.2500    -0.3750
```

```
f = 3x3
    -0.4444    -0.1111    -0.4444
    -0.1111     3.2222    -0.1111
    -0.4444    -0.1111    -0.4444
```



高增幅濾波器 (High Boost Filter)

High boost filter 跟 unsharpening filter 要達到的效果一樣，只是 high boost filter 會讓原圖再多乘上倍率，使得圖的邊緣更加凸顯出來。因此公式即為

$$HBF = k(Original) - LPF$$

其中 HBF 表 high boost filter image

LBF 表原圖經 Low Pass filter 變換後的影像

k 為縮放常數

當 $k = 1$ 時即稱 unsharpening mask, 當 $k > 1$ 時即為 high boost filter

最大 & 最小濾波器 (Maximum & Minimum Filter)

最大濾波器的 mask 會把在 mask 內的灰階值先排序, 然後 output 值會是灰階值中最大的, 最小濾波器亦然。因此最大濾波器可以取出影像中每個 pixel 較亮的部分, 最小濾波器反之。

```
img = imread("Image\iguana.png");
img_max = nlfilter(img, [3, 3], "max(x(:))"); % maximum filter
img_min = nlfilter(img, [3, 3], "min(x(:))");
figure;
subplot(1, 2, 1); imshow(img_max);
subplot(1, 2, 2); imshow(img_min);
```



也可以使用中間值濾波器, 而中間值濾波器可以拿來做為去噪點濾波器

```
img = rgb2gray(imread("Image\monkey noise.png"));
```

Warning: PNG library warning: bKGD: invalid.

```
figure; imshow(img)
```



可以看到原圖用許多的黑色噪點，此時如果使用 Median Filter 的話

```
img_med = ordfilt2(img, 5, ones(3, 3));  
figure; imshow(img_med);
```



可以看到黑色噪點被成功移除，原理就是因為噪點會是灰階值最小的，而 median filter 就可以把 mask 裡灰階值最小的過濾掉。

Kuwahara Filter

這個 filter 跟 unsharpening mask 的功能類似，一樣是讓低頻的部分模糊，同時保留 edges 的部分。計算方法是會將 mask 分成四個 submask，並且計算裡面 graylevel 的變異數，output 為變異數最小的 subarea 的 graylevel 平均值

```
img = imread("Image\cameraman.png");
imgm = imfilter(img, ones(3) / 9, "symmetric");
img2s = imfilter(img.^2, ones(3) / 9, "symmetric");
imgv = img2s - imgm.^2;
new_img = zeros(254);
for i = 2:255
    for j = 2:255
        v = [imgv(i-1, j-1) imgv(i-1, j +1) imgv(i+1, j-1) imgv(i+1, j+1)];
        m = [imgm(i-1, j-1) imgm(i-1, j +1) imgm(i+1, j-1) imgm(i+1, j+1)];
        [mi, inx] = min(v);
        out = m(inx);
        new_img(i, j) = out;
    end
end
imshow(uint8(new_img))
```



Domain Filter & Range Filter

前面提到的濾波器都是屬於域濾波器 (Domain Filter)，就是 output 值是直接用 mask 上面的係數乘上對應的灰階值相加而成的。而範圍濾波器 (Range Filter) 則是考慮了像素之間的像素值差距計算而成的。考慮下面的 neighborhood 上執行 Gaussian range filter

0.8	0.1	0.6
0.3	0.5	0.7
0.4	0.9	0.2

周圍 8 個 pixel 值和中間的 pixel 的差值為

0.3	-0.4	0.1
-0.2	0.0	0.2
-0.1	0.4	-0.3

將其值代入 Gaussian Function

$$f(x) = \exp\left\{-\frac{v^2}{2\sigma^2}\right\}$$

在不同的 σ 下可以得到以下值

$\sigma_\gamma = 0.1$:

0.01111	0.00034	0.60653
0.13534	1.00000	0.13534
0.60653	0.00034	0.01111

$\sigma_\gamma = 1$:

0.95600	0.92312	0.99501
0.98020	1.00000	0.98020
0.99501	0.92312	0.95600

$\sigma_\gamma = 10$:

0.99955	0.99920	0.99995
0.99980	1.00000	0.99980
0.99995	0.99920	0.99955

可以看到當 σ 值越大，濾波器會讓影像變得更平緩

Bilateral Filter (雙邊濾波器)

bilateral filter 就是同時把 domain filter 跟 range filter 一起使用，同時考量了 pixel 之間的 closeness 跟 similarity。下面時做一個 bilateral filter，而 domain filter 跟 range filter 都使用 Gaussian Filter。在調整兩個 Gaussian Filter 的標準差值下，我們可以創造出不同的 blurring effect

```
arg = [5 2 0.2;  
       7 10 0.2;  
       11 3 0.1;  
       11 5 0.5];  
  
for i = 1:4  
    subplot(2, 2, i);  
    imgb = bilateral(img, arg(i, 1), arg(i, 2), arg(i, 3));  
    imshow(imgb)  
end
```



Region of Interest

如果只需要對影像的部分區域進行濾波，可以先匡出該範圍，然後再進行濾波

```
img = imread("Image\monkey.png");  
img2 = img(56:281, 221:412);  
  
% 嘗試對猴子的頭進行各種濾波  
xi = [60 27 14 78 130 139];  
yi = [14 38 127 177 160 69];  
roi = roipoly(img2, yi, xi);  
  
a1 = fspecial("average", 10);  
roi1 = roifilt2(a1, img2, roi);  
  
a2 = fspecial("unsharp");  
roi2 = roifilt2(a2, img2, roi);
```

```

a3 = fspecial("log");
roi3 = roifilt2(a3, img2, roi);

a4 = fspecial("gaussian", [25, 25], 10);
roi4 = roifilt2(a4, img2, roi);

figure;
subplot(2, 2, 1); imshow(roi1);
subplot(2, 2, 2); imshow(roi2);
subplot(2, 2, 3); imshow(roi3);
subplot(2, 2, 4); imshow(roi4);

```



```

function out = bilateral(img, w, sd, sr)
    im = im2double(img);
    [r, c] = size(im);
    out = zeros(r, c);
    A = padarray(im, [w, w], 'symmetric');
    G = fspecial('gaussian', 2*w+1, sd);
    for i = 1+w:r+w-1
        for j = 1+w:c+w-1
            R = A(i-w:i+w, j-w:j+w);
            H = exp(-(R-A(i,j)).^2/(2*sr^2));
            F = H.*G;
            out(i-w,j-w) = sum(F(:).*R(:)) / sum(F(:));
        end
    end
end

```

