# 0516212 吳子涵 影像處理 作業一

## (A) Introduction

六張圖片都有各自的不足之處，所以要透過不同的影像處理方式來調整成較好的圖片。

我所使用的方法有下列幾項：

1. contrast adjustment:
    (a) histogram equalization
    (b) log transformation
    (c) power-law transformation

2. color correlation
    (a) color correlation matrix
    (b) Auto White Balance Correction

3. noise reduction
    (a) box filter
    (b) weighted average filter
    (c) median filter
    (d) bilateral filter

4. sharpening
    (a) Laplacian filter4: [0 -1 0; -1 4 -1; 0 -1 0]
    (b) Laplacian filter8: [-1 -1 -1; -1 8 -1; -1 -1 -1]
    (c) Sobel filter
    (d) Robert filter

## (B) Methods

1. contrast adjustment:

(a) **histogram equalization**

step1 先將 RGB 格式的圖片轉換成 Lab，再針對 intensity channel 進行 histogram equalization。

step2 建立一個 count[101]的陣列，index 為 intensity+1（因為 intensity 的值為 0~100，但是陣列的 index 是從 1 開始），所存的值為整張圖中是這個 intensity 的 pixel 個數。將圖片中每個 pixel 的 intensity 四捨五入至整數，然後存在對應的 count 中。

step3 累加 count 中的值，存在 cdf 這個陣列中，並 normalize cdf。

step4 逐一將每個 pixel 的 intensity 四捨五入後，當成 index 來查找 cdf 的值，乘以 100 後即為 histogram equalization 後的 intensity 數值。

step5 從 Lab 格式轉回 RGB。

(b) **log transformation**

step1 將原本圖片的數值從 0~255 正規化至 0~1。

step2 設定常數 c。

step3 利用 c*log(1+ image) 這個公式即可得出經過 log transformation 轉換的圖片。

(c) **power-law transformation**

step1 將原本圖片的數值從 0~255 正規化至 0~1。

step2 設定常數 c, gamma。

step3 利用 c*(image).^gamma 這個公式即可得出經過 power-law transformation 轉換的圖片。

2. color correlation

(a) **color correlation matrix**

step1 將原本圖片的數值從 0~255 正規化至 0~1。

step2 設定顏色轉換矩陣，將這個矩陣分別乘上每個 pixel 的[R;G;B] column，即可得出顏色轉換後的圖片。

(b) **Auto White Balance Correction**

step1 為了估計圖像中外界光源的強度，將 R,G,B 三個 channel 經過 EstimatelIluminantGrey 這個函數處理：

step1-1 計算累計直方圖。

step1-2 當累積到自己所設定的百分比值(p)時，就把它當作光源強度。

step2 建立 EstimateCCT 這個函數來估計色溫。

step3 估計基準光（白光）。

step4 利用 ComputeGainFactorMatrix 這個函數來分別計算基準光與 R,G,B 三個 channel 的比值。

step5 利用 ComputeOffsetMatrix 這個函數來計算偏移量。

step6 利用 step4 與 step5 所得出的矩陣來做白平衡。

3. noise reduction

(a) **box filter**

step1 將原本圖片的數值從 0~255 正規化至 0~1。

step2 建立 convolve 函數：

step2-1 利用 rot90(kernel, 2) 將 kernel 翻轉。

step2-2 為了在轉換後能保持與原圖一樣的大小，將原本圖片做 zero padding。

step2-3 將 kernel 的中心逐一對在原圖的每個 pixel 上，並將原圖在 kernel 範圍內的數值乘上相對應的 kernel 內的值，加總後即可達成 convolution 的效果。

step3 kernel 為[1,1,1; 1,1,1; 1,1,1]/9，分別與 R,G,B 三個 channel 經過 convolve 這個函數做 convolution，即可得出 box filter 的效果。

(b) **weighted average filter**

step1 將原本圖片的數值從 0~255 正規化至 0~1。

step2 kernel 為[1 2 1; 2 4 2; 1 2 1]/16，分別與 R,G,B 三個 channel 經過 convolve 這個函數做 convolution（函數同上），即可得出 weighted average filter 的效果。

(c) **median filter**

step1 將原本圖片的數值從 0~255 正規化至 0~1。

step2 建立 median 函數：

step2-1 設定變數 x，而 kernel size 為 2*x+1。

step2-2 為了在轉換後能保持與原圖一樣的大小，將原本圖片做 zero padding。

step2-3 將 kernel 的中心逐一對在原圖的每個 pixel 上，並紀錄原圖在 kernel 範圍內的數值，再執行排序，最終取中間值作為轉換後的數值。

step3 分別將 R,G,B 三個 channel 經過 median 這個函數。

(d) **bilateral filter**

step1 將原本圖片的數值從 0~255 正規化至 0~1。

step2 將 RGB 格式的圖片轉換成 Lab。

step3 設定此高斯濾波器的權重：G 代表距離的權重、H 代表像素色差的權重。

step4 F = H*G 為最終的高斯濾波器權重。

step5 逐一將每個 pixel 的 R,G,B channel 分別帶入高斯濾波的轉換公式，即可得到 bilateral filter 的效果。

4. sharpening

(a) **Laplacian filter4**

step1 將 RGB 格式的圖片轉換成 Lab。

step2 kernel 為[0 -1 0; -1 4 -1; 0 -1 0]，與原圖的 intensity channel 當成參數，執行 convolve 函數（函數同上），最後會得到邊界。

step3 將邊界加在原圖上，即可達到銳化的效果。

step4 將 Lab 格式的圖片轉換成 RGB。

(b) **Laplacian filter8**

step1 將 RGB 格式的圖片轉換成 Lab。

step2 kernel 為[-1 -1 -1; -1 8 -1; -1 -1 -1]，與原圖的 intensity channel 當成參數，執行 convolve 函數（函數同上），最後會得到邊界。

step3 將邊界加在原圖上，即可達到銳化的效果。

step4 將 Lab 格式的圖片轉換成 RGB。

(c) **Sobel filter**

step1 將 RGB 格式的圖片轉換成 Lab。

step2 kernel 為[-1 -2 -1; 0 0 0; 1 2 1]，與原圖的 intensity channel 當成參數，執行 convolve 函數（函數同上），最後會得到邊界。

step3 將邊界加在原圖上，即可達到銳化的效果。

step4 將 Lab 格式的圖片轉換成 RGB。

(d) **Robert filter**

step1 將 RGB 格式的圖片轉換成 Lab。

step2 kernel 為[-1 0; 0 1]，與原圖的 intensity channel 當成參數，執行 convolve 函數（函數同上），最後會得到邊界。

step3 將邊界加在原圖上，即可達到銳化的效果。

step4 將 Lab 格式的圖片轉換成 RGB。

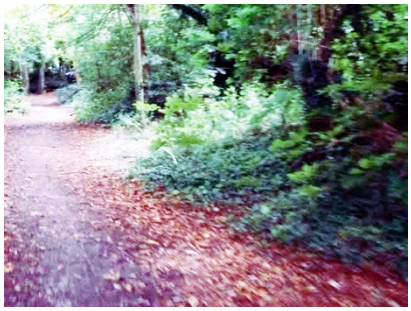
## (C) Explanation of the experiments and result

(左、右圖分別為處理前、處理完的圖，放旁邊以方便比較)

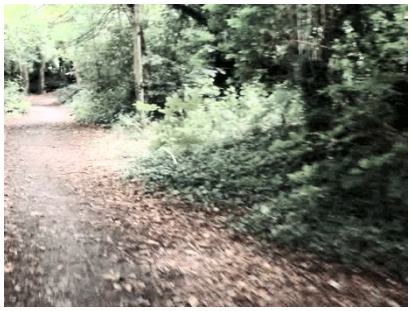
for p1im1:

Step1 contrast adjustment

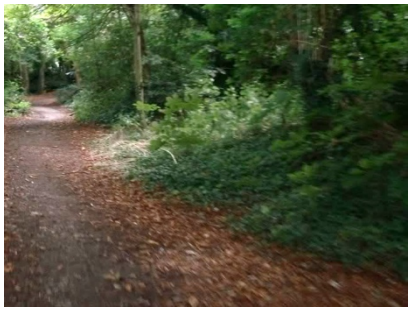
method 1: 將 r, g, b 三個 channel 分別做 histogram equalization，但發現結果的色調會與原本的圖片不一樣。

method 2: 將 color space 從 RGB 轉到 Lab，只將 intensity 的 channel 做 histogram equalization，雖然色調能夠保持不變，但圖片會變得太亮，呈現類似於過曝的狀況。

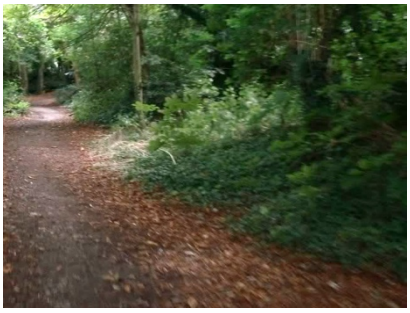



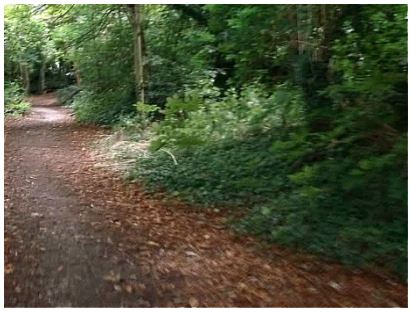
method 3: 因為圖片整體偏暗，所以我使用 log transformation，突顯暗處之細節。

Step2 sharpening

利用 Robert filter 來做影像銳化。





for p1im2:

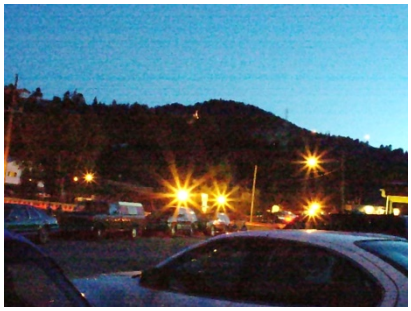
Step1 contrast adjustment

因為圖片整體偏亮，所以我使用 power-law transform，突顯亮處之細節。

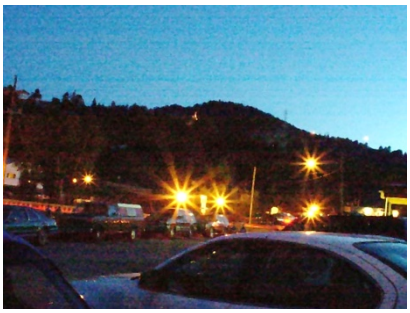
Step2 我有試著使用不同的銳化方式來處理圖片，也試過先使用 median filter 再銳化，但效果都不比上圖好。

for p1im3:

Step1 因為圖片整體偏暗，所以我使用 log transformation，突顯暗處之細節。(c=1.6)



Step2 將 R,G,B 三個 channel 分別利用 median filter 來去除雜訊。




Step3 利用 Robert filter 進行邊界銳化。


for p1im4:

Step1 從路標中的文字顏色得知，整張圖片的色調偏藍，故我使用 Auto White Balance Correction 來修正色調。(參數=96)



Step2 因為在背景的白色區域有很多其他顏色的雜訊，所以我使用 weighted average kernel 來執行平滑化。


Step3 使用 Robert filter 來銳利化邊界。



Step4 使用 log transformation 來調亮圖片。(常數 c 為 1.6)



for p1im5:

Step1 照片整體稍微偏藍，所以我利用 Auto White Balance Correction 來修正色調 (參數=90)。

Step2 利用 log transformation 來提高亮度 (係數 c 為 1.8)。

Step3 銳化邊界

method1: 利用 Robert filter 來銳化邊界。


天空白色的地方會因為有其他雜訊而產生凹凸不平效果，故改採用下列的方式。

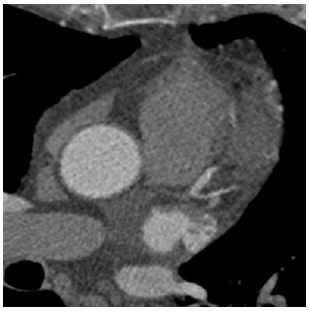
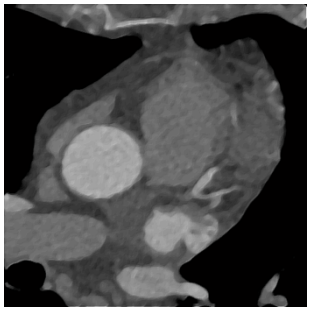
method2: 先使用 weighted average filter 平滑化圖片。


再使用 Laplacian filter 來銳化邊界。

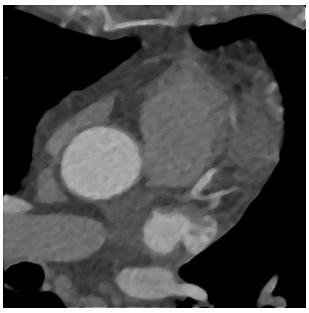
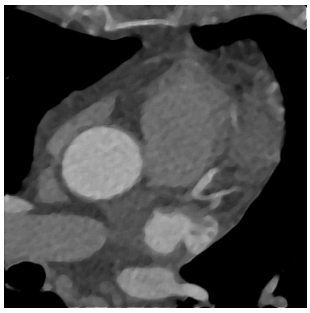

for p1im6:

method1

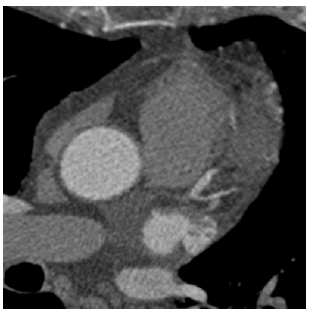
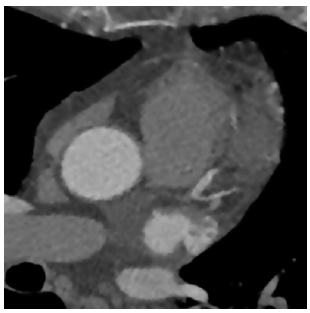
Step1: 因為這張圖的雜訊比較像 pepper and salt，所以將 R,G,B 三個 channel 分別利用 median filter 來去除雜訊。

Step2: 使用完 median filter 之後影像會變模糊，所以我再使用 Robert filter 來銳化邊界。

method2 使用 bilateral filter 來同時消除雜訊，也保留相當程度的邊界資訊。

## (D) Discussion

Observation

在銳化邊界的方面，我將每張圖分別套用上述四種不同的 filter 來處理，最終選擇一個我認為處理得較好的圖。在這個過稱中發現，比較常選擇的是 Robert filter，其次是 Laplacian filter，而都沒有選擇 Sobel filter 的原因是因為他會讓整張圖看起來過於立體，呈現不自然的狀況。

在調整色調的部分，一開始使用 color correction matrix，但是矩陣內的參數要自己慢慢調，我調了很久還是沒辦法呈現出我滿意的色調，所以我改用 AWB 來調整色調。

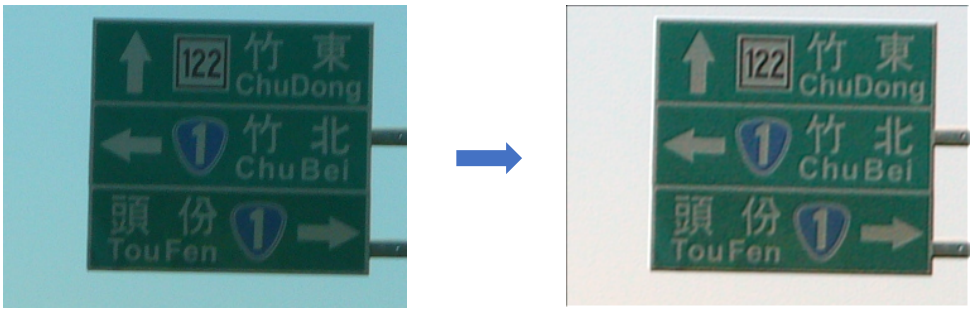
Result & Remaining Question

for p1im1: 放大整張圖會發現有一些雜訊，但無法使用平滑濾波器來消除雜訊，否則會丟失細節。

for p1im2: 因為細節太小導致不易強調細節。

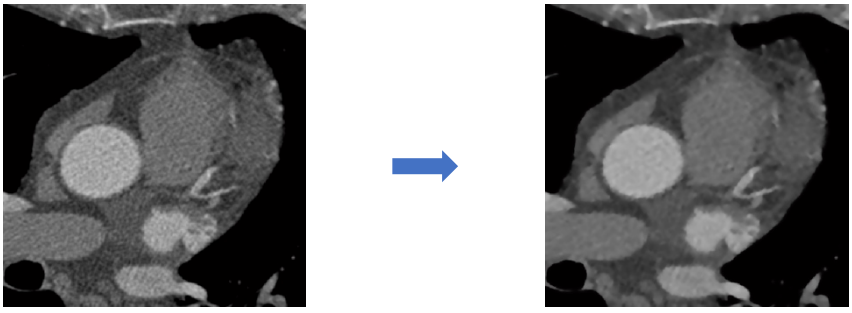
for p1im3: 因為同時存在非常黑與非常白的值，導致不易調整 intensity。另一方面，因為天空的雜訊很多，無法有效地消除，選擇的處理方式要權衡雜訊的處理與細節的保留。

for p1im4: 在使用銳利化邊界的同時，綠色板子上會因為有一些雜訊而呈現稍微凸起的狀況。




for p1im5: 因為細節太小，在銳利化邊界時較無法有效地強調細節。

for p1im6: 因為原本的雜訊過多，所以處理完仍然有一些雜訊。



## (E) Program Code

Histogram Equalization

```
function hist_im_I = histogram_intensity(origin_image)

[h,w] = size(origin_image(:,:,1));
lab_im = rgb2lab(origin_image);
count=zeros(1,101);

for i = 1:h
    for j = 1:w
        count(round(lab_im(i,j,1))+1) = count(round(lab_im(i,j,1))+1) + 1;
    end
end

cdf = cumsum(count);
cdf = cdf / numel(lab_im(:,:,1));

for i = 1:h
    for j = 1:w
        lab_im(i,j,1) = cdf(round(lab_im(i,j,1))+1)*100;
    end
end

hist_im_I= lab2rgb(lab_im);

end
```

Log Transformation

```
function log_im = logarithm(origin_image)

double_image = im2double(origin_image);
c = 1.8;
log_im = c*log(1+double_image);

end
```

Power-Law Transformation

```
function power_im = power_law(origin_image)

double_image = im2double(origin_image);
c = 1;
gamma = 3;
power_im = c*(double_image).^gamma;

end
```

## Color Correction Matrix

```
function ccm_im = ccm(origin_image)

double_image = im2double(origin_image);

[h,w] = size(double_image(:,:,1));

final_image = zeros(h,w);
for i = 1:h
    for j = 1:w

        tmp = [1.1 0 0;
               0.2 1.1 -0.2;
               0  0  1.1] * [double_image(i,j,1);double_image(i,j,2);double_image(i,j,3)];

        final_image(i,j,1) = tmp(1,1);
        final_image(i,j,2) = tmp(2,1);
        final_image(i,j,3) = tmp(3,1);

    end
end

ccm_im = final_image;
end
```

## Auto White Balance

```
function O = AWB(I, p)

if nargin == 1
    p = 96;
end
p = 100-p;
szI = size(I);

[iR, iG, iB] = EstimateIlluminantRGB(I, p);

iEstm = [iR, iG, iB];
CCT_Estm = EstimateCCT(iEstm);
iRef = [iG, iG, iG];
CCT_Ref = EstimateCCT(iRef);

K = ComputeGainFactorMatrix(iEstm);
T = ComputeOffsetMatrix(K, CCT_Estm, CCT_Ref);

O = PerformWhiteBalanceCorrection(I, K, T);

end

function [Rc, Gc, Bc] = EstimateIlluminantRGB(I, p)
    R = I(:,:,1);
    G = I(:,:,2);
    B = I(:,:,3);

    Rc = EstimateIlluminantGrey(R, p);
    Gc = EstimateIlluminantGrey(G, p);
    Bc = EstimateIlluminantGrey(B, p);
end
```

```
function Ic = EstimateIlluminantGrey(I, p)
    Ic = 0;

    L = 256;
    sz = size(I);
    pxlTh = (p*sz(1)*sz(2))/100;
    histI = histogram(I);
    Imin = min(min(I));
    Imax = max(max(I));

    for k=Imin:(Imax-1)

        j = double(k+1);
        cnt1 = sum(histI(j:L));

        j = j+1;
        cnt2 = sum(histI(j:L));
        if( (cnt1 > pxlTh) && (cnt2 < pxlTh) )
            Ic = k;
             break;
        end
    end
end

function CCT = EstimateCCT(iEstm)

    A0 = -949.86315;
    A1 = 6253.80338;
    A2 = 28.70599;
    A3 = 0.00004;
    t1 = 0.92159;
    t2 = 0.20039;
    t3 = 0.07125;
    xe = 0.3366;
    ye = 0.1735;

    XYZ_Conv_matrix = [ 0.4124 0.3576 0.1805;
                        0.2126 0.7152 0.0722;
                        0.0193 0.152 0.9505];

    XYZ = XYZ_Conv_matrix * double(iEstm');

    x = XYZ(1) / (sum(XYZ));
    y = XYZ(2) / (sum(XYZ));
    H = -((x-xe)/(y-ye));
    CCT = A0 + (A1*exp(H/t1)) + (A2*exp(H/t2)) + (A3*exp(H/t3));
end

function [K] = ComputeGainFactorMatrix(iEstm)

    iEstm = double(iEstm);
    iEstm_R = iEstm(1);
    iEstm_G = iEstm(2);
    iEstm_B = iEstm(3);
    iRef_R = iEstm_G;
    iRef_G = iEstm_G;
    iRef_B = iEstm_G;
    Kr = iRef_R / iEstm_R;
    Kg = iRef_G / iEstm_G;
    Kb = iRef_B / iEstm_B;
    K = [Kr, 0, 0;
          0, Kg, 0;
          0, 0, Kb];
end
```

```
function T = ComputeOffsetMatrix(K, CCT_Estm, CCT_Ref)
    A = 100;
    Kr = K(1,1);
    Kb = K(3,3);
    Tr = max(1, (CCT_Estm - CCT_Ref)/A ) * (Kr-1);
    Tg = 0;
    Tb = max(1, (CCT_Ref - CCT_Estm)/A ) * (Kb-1);
    T = [Tr; Tg; Tb];
end

function O = PerformWhiteBalanceCorrection(I, K, T)
    sz = size(I);
    O = uint8(zeros(sz));

    for x = 1:sz(1)
        for y = 1:sz(2)
            Fxy = double([I(x,y,1), I(x,y,2), I(x,y,3)]');
            FWB = K * Fxy + T;

            for p=1:3
                O(x,y,p) = uint8(FWB(p));
            end
        end
    end
end

function hist = histogram(image)

[h,w] = size(image);
count=zeros(1,256);
for i = 1:h
    for j = 1:w
        count(image(i,j)+1) = count(image(i,j)+1) + 1;
    end
end

cdf = cumsum(count);
cdf = cdf / numel(image(:,:));

final_image_r=zeros([h,w,1]);

for i = 1:h
    for j = 1:w
        final_image(i,j) = cdf(image(i,j)+1);
    end
end

subplot(1,2,2);
imshow(final_image);
hist = count;

end
```

## Box Filter & Weighted Average Filter

```
function smooth_im = smooth(origin_image)

double_image = im2double(origin_image);
kernel = ones(3,3)/9;
final_image_1(:,:,1) = convolve(double_image(:,:,1),kernel);
final_image_1(:,:,2) = convolve(double_image(:,:,2),kernel);
final_image_1(:,:,3) = convolve(double_image(:,:,3),kernel);

kernel = [1 2 1; 2 4 2; 1 2 1]/16;
final_image_2(:,:,1) = convolve(double_image(:,:,1),kernel);
final_image_2(:,:,2) = convolve(double_image(:,:,2),kernel);
final_image_2(:,:,3) = convolve(double_image(:,:,3),kernel);

smooth_im = final_image_2;

end

function B = convolve(A, k)
[r,c] = size(A);
[m,n] = size(k);
h = rot90(k, 2);
center = floor((size(h)+1)/2);
left = center(2) - 1;
right = n - center(2);
top = center(1) - 1;
bottom = m - center(1);
Rep = zeros(r + top + bottom, c + left + right);
for x = 1 + top : r + top
    for y = 1 + left : c + left
        Rep(x,y) = A(x - top, y - left);
    end
end
B = zeros(r , c);
for x = 1 : r
    for y = 1 : c
        for i = 1 : m
            for j = 1 : n
                q = x - 1;
                w = y -1;
                B(x, y) = B(x, y) + (Rep(i + q, j + w) * h(i, j));
            end
        end
    end
end
end
```

## Median Filter

```
function median_rgb_im = median_rgb(origin_image)

double_image = im2double(origin_image);
final_image(:,:,1) = median(double_image(:,:,1));
final_image(:,:,2) = median(double_image(:,:,2));
final_image(:,:,3) = median(double_image (:,:,3));

median_rgb_im = final_image;

end
```

```
function med_im = median(origin_image)

[h,w] = size(origin_image(:,:,1));
x = 2;
filter_size = 2*x+1;
padding_image = zeros(size(origin_image)+2*x);
median_image = zeros(size(origin_image));

for i = 1:h
    for j = 1:w
        padding_image(i+x,j+x) = origin_image(i,j);
    end
end

for i = 1:h
    for j = 1:w
        tmp = zeros(9,1);
        idx=1;
        for p = 1:filter_size
            for q = 1:filter_size
                tmp(idx) = padding_image(i+p-1, j+q-1);
                idx = idx + 1;
            end
        end
        med = sort(tmp);
        median_image(i,j) = med(floor(filter_size*filter_size/2)+1);
    end
end
med_im = median_image;
end
```

Bilateral Filter

```
function B = bilateral(A,w,sigma)
A = im2double(A);
if size(A,3) == 1
   B = bfltGray(A,w,sigma(1),sigma(2));
else
   B = bfltColor(A,w,sigma(1),sigma(2));
end
end

function B = bfltGray(A,w,sigma_d,sigma_r)
[X,Y] = meshgrid(-w:w,-w:w);
G = exp(-(X.^2+Y.^2)/(2*sigma_d^2));
dim = size(A);
B = zeros(dim);
for i = 1:dim(1)
   for j = 1:dim(2)

         iMin = max(i-w,1);
         iMax = min(i+w,dim(1));
         jMin = max(j-w,1);
         jMax = min(j+w,dim(2));
         I = A(iMin:iMax,jMin:jMax);
         H = exp(-(I-A(i,j)).^2/(2*sigma_r^2));

         F = H.*G((iMin:iMax)-i+w+1,(jMin:jMax)-j+w+1);
         B(i,j) = sum(F(:).*I(:))/sum(F(:));

   end
end
end
```

```
function B = bfltColor(A,w,sigma_d,sigma_r)
if exist('applycform','file')
   A = applycform(A,makecform('srgb2lab'));
else
   A = colorspace('Lab<-RGB',A);
end
[X,Y] = meshgrid(-w:w,-w:w);
G = exp(-(X.^2+Y.^2)/(2*sigma_d^2));
sigma_r = 100*sigma_r;

dim = size(A);
B = zeros(dim);
for i = 1:dim(1)
   for j = 1:dim(2)

         iMin = max(i-w,1);
         iMax = min(i+w,dim(1));
         jMin = max(j-w,1);
         jMax = min(j+w,dim(2));
         I = A(iMin:iMax,jMin:jMax,:);

         dL = I(:,:,1)-A(i,j,1);
         da = I(:,:,2)-A(i,j,2);
         db = I(:,:,3)-A(i,j,3);
         H = exp(-double(dL.^2+da.^2+db.^2)/double(2*sigma_r^2));

         F = H.*G((iMin:iMax)-i+w+1,(jMin:jMax)-j+w+1);

         I = double(I);
         norm_F = sum(F(:));

         B(i,j,1) = sum(sum(F.*I(:,:,1)))/norm_F;
         B(i,j,2) = sum(sum(F.*I(:,:,2)))/norm_F;
         B(i,j,3) = sum(sum(F.*I(:,:,3)))/norm_F;
   end
end
if exist('applycform','file')
   B = applycform(B,makecform('lab2srgb'));
else
   B = colorspace('RGB<-Lab',B);
end
end
```

Sharpening Filter

```
function sharpen_im = sharpen(origin_image)

lab_im = rgb2lab(origin_image);
kernel = [-1 -1 -1; -1 8 -1; -1 -1 -1];
lab_im(:,:,1) = lab_im(:,:,1) + convolve(double(lab_im(:,:,1)), kernel);
final_image_1 = lab2rgb(lab_im);

lab_im = rgb2lab(origin_image);
kernel = [0 -1 0; -1 4 -1; 0 -1 0];
lab_im(:,:,1) = lab_im(:,:,1) + convolve(double(lab_im(:,:,1)), kernel);
final_image_2 = lab2rgb(lab_im);

lab_im = rgb2lab(origin_image);
kernel = [-1 -2 -1; 0 0 0; 1 2 1];
lab_im(:,:,1) = lab_im(:,:,1) + convolve(double(lab_im(:,:,1)), kernel);
final_image_3 = lab2rgb(lab_im);

lab_im = rgb2lab(origin_image);
kernel = [-1 0; 0 1];
lab_im(:,:,1) = lab_im(:,:,1) + convolve(double(lab_im(:,:,1)), kernel);
final_image_4 = lab2rgb(lab_im);

sharpen_im = final_image_4;
end

function B = convolve(A, k);
[r c] = size(A);
[m n] = size(k);
h = rot90(k, 2);
center = floor((size(h)+1)/2);
left = center(2) - 1;
right = n - center(2);
top = center(1) - 1;
bottom = m - center(1);
Rep = zeros(r + top + bottom, c + left + right);
for x = 1 + top : r + top
    for y = 1 + left : c + left
        Rep(x,y) = A(x - top, y - left);
    end
end
B = zeros(r , c);
for x = 1 : r
    for y = 1 : c
        for i = 1 : m
            for j = 1 : n
                q = x - 1;
                w = y -1;
                B(x, y) = B(x, y) + (Rep(i + q, j + w) * h(i, j));
            end
        end
    end
end
end
```