MATLAB CODE FOR UNDERWATER IMAGE ENHANCEMENT USING COLOR BALANCE AND FUSION

CODE:

```
clc;
clear all;
close all;

file = imgetfile;
if file == 0
warndlg('please select input image')
return
else
in = imread(file);
figure('name','Input Image');
imshow(in);
end

% white balance
img1 = ColorBalance(in);
cform = makecform('srgb2lab');
lab1 = applycform(img1,cform);
figure, imshow(img1);

% median filter
a = double(img1);
b = a;
[row, col] = size(a);

%MSE and PSNR measurement
[row, col] = size(in);
mse = sum(sum((in(1,1) - b(1,1)).^2)) / (row * col);
psnr = 10 * log10(255 * 255 / mse);

disp('<--------------- Median filter ---------------------------->');
disp('Mean Square Error ');
disp(mse);
disp('Peak Signal to Noise Ratio');
disp(psnr);
disp('<-------------------------------------------------------->');
for x = 2:1:row-1
for y = 2:1:col-1
%% To make a 3x3 mask into a 1x9 mask
a1 = [a(x-1,y-1) a(x-1,y) a(x-1,y+1) a(x,y-1) a(x,y) a(x,y+1)...
a(x+1,y-1) a(x+1,y) a(x+1,y+1)];
a2 = sort(a1);
med = a2(5); % the 5th value is the median
b(x,y) = med;
```

```matlab
end
end
figure(1); imshow(uint8(img1))
figure(2); imshow(uint8(b))

% CLAHE
lab2 = lab1;
lab2(:, :, 1) = adapthisteq(lab2(:, :, 1));
cform = makecform('lab2srgb');
img2 = applycform(lab2,cform);
figure,imshow(img2);

% input
R1 = double(lab1(:, :, 1)) / 255;
R2 = double(lab2(:, :, 1)) / 255;


% calculate laplacian contrast weight
Wlap1 = abs(imfilter(R1, fspecial('Laplacian'), 'replicate', 'conv'));
figure, imshow(Wlap1);
Wlap2 = abs(imfilter(R2, fspecial('Laplacian'), 'replicate', 'conv'));
figure, imshow(Wlap2);
%calculate Local contrast weight
h = 1/16* [1, 4, 6, 4, 1];
Wcont1 = imfilter(R1, transpose(h)*h, 'replicate', 'conv');
Wcont1 = (R1 - Wcont1).^2;

Wcont2 = imfilter(R2, transpose(h)*h, 'replicate', 'conv');
Wcont2 = (R2 - Wcont2).^2;


% calculate the saliency weight
Wsal1 = saliency_detection(img1);

Wsal2 = saliency_detection(img2);


% calculate the exposedness weight
sigma = 0.25;
avg = 0.5;
Wexp1 = exp(-(R1 - avg).^2 / (2*sigma^2));
Wexp2 = exp(-(R2 - avg).^2 / (2*sigma^2));


% calculate the normalized weight
W1 = (Wlap1 + Wcont1 + Wsal1 + Wexp1) ./ (Wlap1 + Wcont1 + Wsal1 + Wexp1 +
Wlap2 + Wcont2 + Wsal2 + Wexp2);
W2 = (Wlap2 + Wcont2 + Wsal2 + Wexp2) ./ (Wlap1 + Wcont1 + Wsal1 + Wexp1 +
Wlap2 + Wcont2 + Wsal2 + Wexp2);
```

```
% calculate the gaussian pyramid
level = 5;
Weight1 = gaussian_pyramid(W1, level);

Weight2 = gaussian_pyramid(W2, level);
close all
% calculate the laplacian pyramid
% input1
R1 = laplacian_pyramid(double(double(img1(:, :, 1))), level);
G1 = laplacian_pyramid(double(double(img1(:, :, 2))), level);
B1 = laplacian_pyramid(double(double(img1(:, :, 3))), level);
% input2
R2 = laplacian_pyramid(double(double(img2(:, :, 1))), level);
G2 = laplacian_pyramid(double(double(img2(:, :, 2))), level);
B2 = laplacian_pyramid(double(double(img2(:, :, 3))), level);

% fusion
for i = 1 : level
r_py{i} = Weight1{i} .* R1{i} + Weight2{i} .* R2{i};
g_py{i} = Weight1{i} .* G1{i} + Weight2{i} .* G2{i};
b_py{i} = Weight1{i} .* B1{i} + Weight2{i} .* B2{i};
end


for i = level : -1 : 2
[m, n] = size(g_py{i - 1});
g_py{i - 1} = g_py{i - 1} + imresize(g_py{i}, [m, n]);
end
G = g_py{1};

for i = level : -1 : 2
[m, n] = size(r_py{i - 1});
r_py{i - 1} = r_py{i - 1} + imresize(r_py{i}, [m, n]);
end
R = r_py{1};


for i = level : -1 : 2
[m, n] = size(b_py{i - 1});
b_py{i - 1} = b_py{i - 1} + imresize(b_py{i}, [m, n]);
end
B = b_py{1};


fusion = cat(3, uint8(R), uint8(G), uint8(B));
figure, imshow(fusion)
figure, imshow([in, fusion])
```