University of Messina

Department of Engineering
Master's Degree Course in Engineering and Computer Science

Multimedia Digital Signal Processing
Project: Enhancing Under Water Images using Image Processing

Student:
Girija Jyothi Golla
Student id-541761
Academic Year- 2023/2024

Teacher:
Prof. Salvatore Serrano

Table of contents

Introduction:

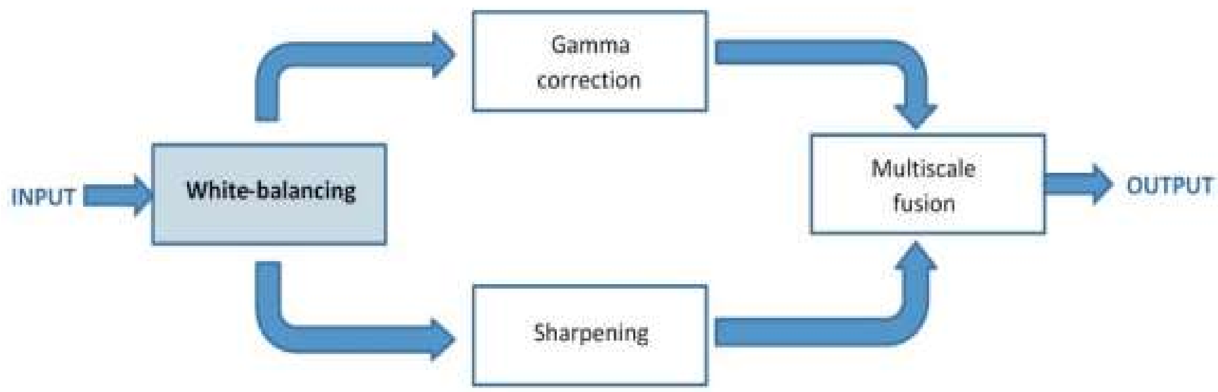
Introduced an effective technique for improving underwater images affected by scattering and absorption, without the need for specialized hardware or prior knowledge of underwater conditions or scene structure. This approach involves a single technique, relying on the fusion of images derived directly from a color-compensated and white-balanced version of the original degraded image. These images, along with their corresponding weight maps, are carefully designed to facilitate the transfer of edges and color contrast to the final output.

To address potential artifacts in the low-frequency components caused by sharp weight map transitions, we incorporate a multiscale fusion strategy. Through comprehensive qualitative and quantitative evaluations, we demonstrate that our enhanced images exhibit enhanced exposure in dark areas, improved overall contrast and sharp edges.

Solution approach:

- Our approach builds first on Gaussian pyramids, constructed by iteratively smoothing and down-sampling the input image using a Gaussian filter, providing a multi-scale representation beneficial for tasks like image analysis and object recognition.
- Laplacian pyramids are formed by computing the difference between each Gaussian pyramid level and its expanded version, effectively decomposing the image into scale-specific details. It excels in capturing high-frequency components, making it valuable for applications like image compression, edge detection and synthesis.
- After that the original image can be accurately reconstructed by combining Laplacian pyramid levels with the last Gaussian pyramid level.
- In the end, use of saliency after image reconstruction depends on the application and goals of the image processing pipeline.

Architecture:



White Balancing: It adjusts the color temperature of an image to ensure accurate representation by removing unwanted color casts.

Gamma correction: It enhances the brightness and contrast of an image by compensation for non-linearities in display devices.

Sharpening: It improves image clarity by emphasizing edges and details, enhancing overall visual perception.

Multi-scale fusion: It combines information from different scales to create a comprehensive representation, beneficial for tasks such as image enhancement and feature extraction.

Dataset:

The EUVP (Enhancing Underwater Visual Perception) dataset contains a separate sets of paired and unpaired image samples of poor and good perceptual quality to facilitate supervised training of underwater image enhancement models.

The dataset is downloaded from this source

https://irvlab.cs.umn.edu/resources/euvp-dataset

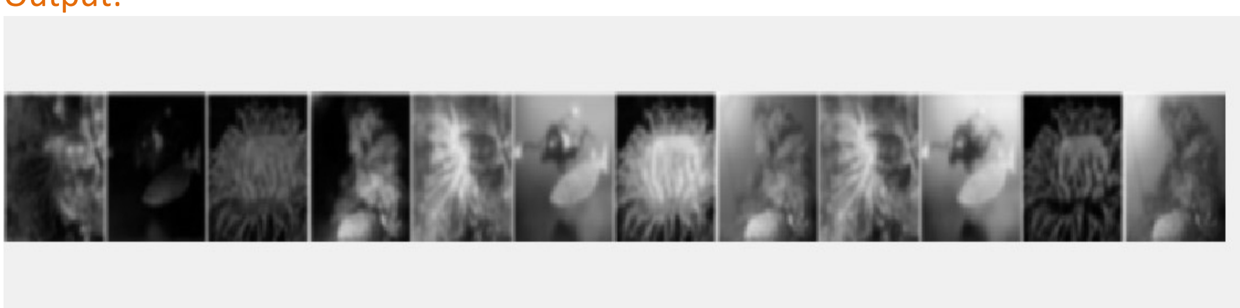
Matlab code with output:

Gaussian

```matlab
function out = gaussian_pyramid(img, level)
h = 1/16* [1, 4, 6, 4, 1]; % creates a 1D Gaussian filter smoothing
filt = h'*h; %computes the 2D filter
out{1} = imfilter(img, filt, 'replicate', 'conv'); %image by replicating the border pixels and convolution
temp_img = img;
for i = 2 : level
    temp_img = temp_img(1 : 2 : end, 1 : 2 : end); % selects every other pixel along each dimension, effectively reducing the image size by half
    out{i} = imfilter(temp_img, filt, 'replicate', 'conv'); % downsampling
end
```

```matlab
% Load the image
img = imread('C:\Users\lenovo\Downloads\img.jpg');  % Replace with the path to your image

% Choose the desired number of pyramid levels
level = 5;

% Call the function to create the pyramid
pyramid = gaussian_pyramid(img, level);

% Display the original image
figure;
imshow(img);

% Display one level of the pyramid (e.g., the second level)
figure;
imshow(pyramid{2});
```

Output:



Laplacian

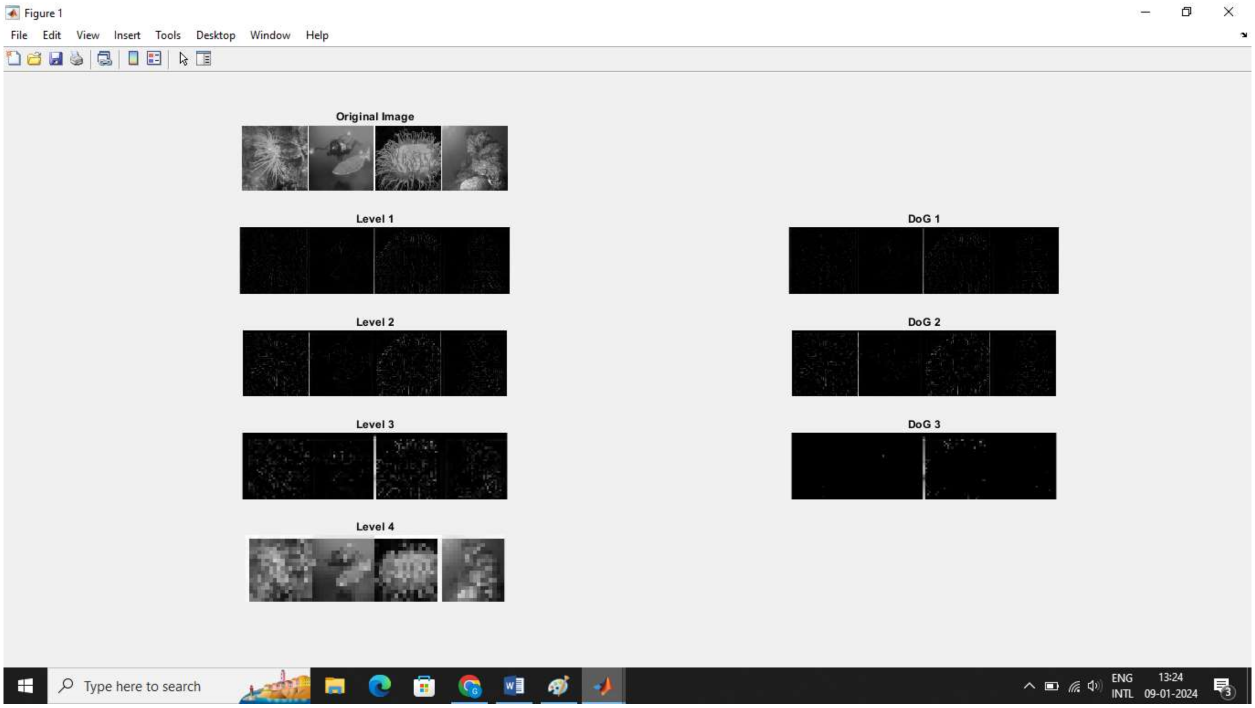
Workspace | Editor - laplacian_pyramid.m

gaussian_pyramid.m | laplacian_pyramid.m | ImageEnhancement.m | pyramid_reconstruct.m | saliency_detection.m | main_script.m | main_script_la

```
function out = laplacian_pyramid(img, level)
h = 1/16* [1, 4, 6, 4, 1];  % Define a 1D Gaussian filter kernel
%filt = h'*h;
out{1} = img;
temp_img = img;  % Initialize a temporary variable to store the current image
for i = 2 : level % Downsample the temporary image by a factor of 2
    temp_img = temp_img(1 : 2 : end, 1 : 2 : end);
    %out{i} = imfilter(temp_img, filt, 'replicate', 'conv');
    out{i} = temp_img;
end
% Calculate the Difference of Gaussians (DoG) between consecutive levels
for i = 1 : level - 1
    [m, n] = size(out{i});  % Resize the next level to match the size of the current level
    out{i} = out{i} - imresize(out{i+1}, [m, n]); % Calculate the DoG by subtracting the resized next level from the current level
end
```

```
img = imread('C:\Users\lenovo\Downloads\img.jpg');  % Replace 'your_image_path.jpg' with the actual image pat

% Convert the image to grayscale if it's a color image
if size(img, 3) == 3
    img = rgb2gray(img);
end

% Choose the number of pyramid levels
pyramid_levels = 4;

% Call the laplacian_pyramid function
pyramid = laplacian_pyramid(img, pyramid_levels);

% Display the original image and the pyramid levels
figure;

subplot(pyramid_levels+1, 2, 1);
imshow(img);
title('Original Image');

for i = 1 : pyramid_levels ...

% Display the difference of Gaussians (DoG)
for i = 1 : pyramid_levels - 1 ...
```

Output:

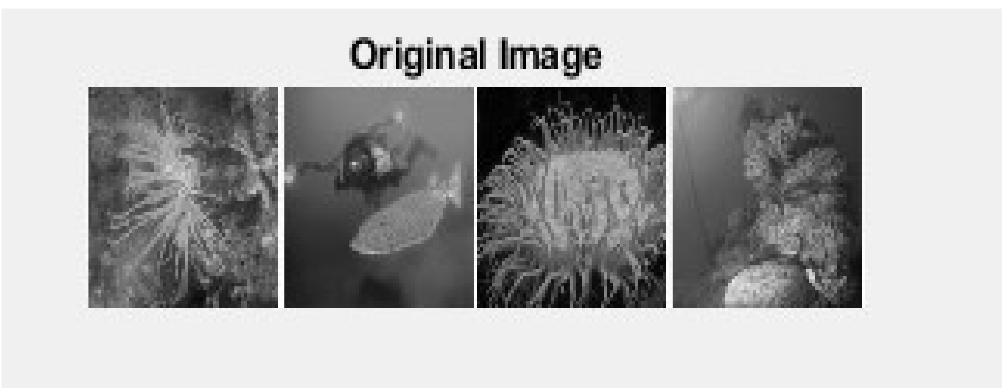


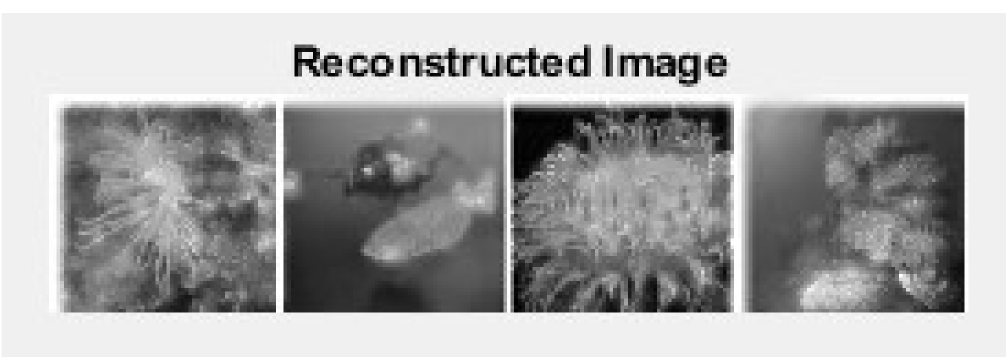
Reconstruction of an image

```
gaussian_pyramid.m | laplacian_pyramid.m | ImageEnhancement.m | pyramid_reconstruct.m | saliency_detection.m | main_script.m | main_script_lap

function out = pyramid_reconstruct(pyramid) % Function to reconstruct an image from a Laplacian pyramid.
level = length(pyramid); % Get the number of levels in the Laplacian pyramid
for i = level : -1 : 2 % Loop through the pyramid levels in reverse order (from the finest scale to the coarsest scale)

    [m, n] = size(pyramid{i - 1});  % Extract size information from the image at the current level - 1

    pyramid{i - 1} = pyramid{i - 1} + imresize(pyramid{i}, [m, n]); %the image from the next finer scale to matchsize current level
end
out = pyramid{1}; % The reconstructed image is the base image at the coarsest scale
```

```
% Example main script for Laplacian pyramid reconstruction

% Load an example image
img = imread('C:\Users\lenovo\Downloads\img.jpg');  % Replace 'your_image_path.jpg' with the actual image path

% Convert the image to grayscale if it's a color image
if size(img, 3) == 3
    img = rgb2gray(img);
end

% Call the laplacian_pyramid function
pyramid = laplacian_pyramid(img, pyramid_levels);

% Reconstruct the image from the Laplacian pyramid
reconstructed_image = pyramid_reconstruct(pyramid);

% Display the original image, pyramid levels, and reconstructed image ...
figure('Position', [100, 100, 2100, 2300]);

subplot(pyramid_levels+2, 2, 1);
imshow(img);
title('Original Image');

% Display the reconstructed image
subplot(pyramid_levels+2, 2, (pyramid_levels+1)*2 - 1);
imshow(reconstructed_image);
title('Reconstructed Image');
```

Output:



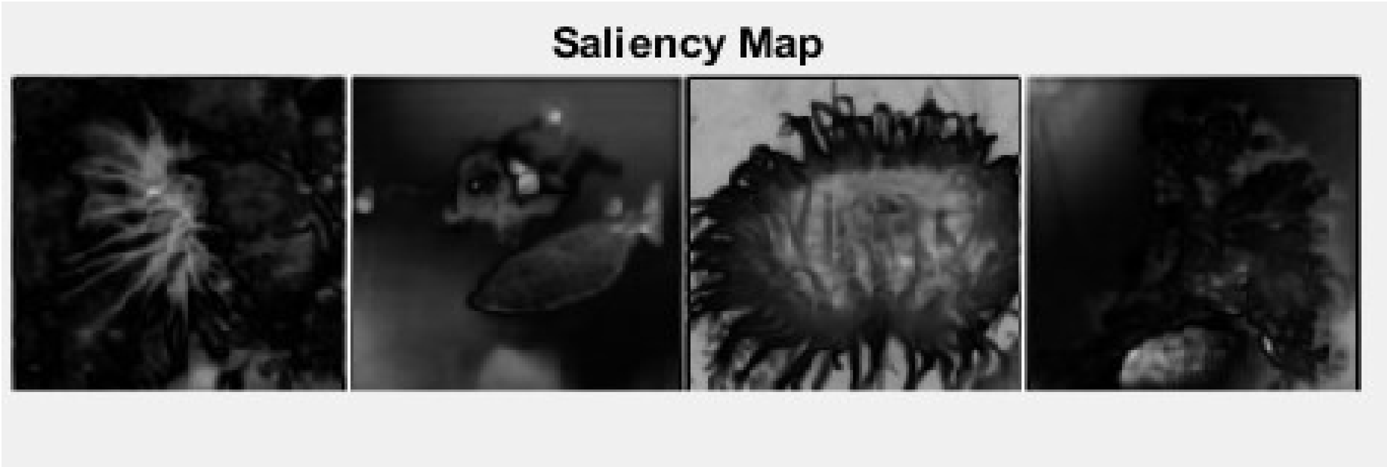


Saliency detection:

```
function sm = saliency_detection(img) % Function to perform saliency detection on an input image.

% Read image and blur it with a 3x3

%img = imread('input_image.jpg');%Provide input image path
gfrgb = imfilter(img, fspecial('gaussian', 3, 3), 'symmetric', 'conv');

cform = makecform('srgb2lab');
lab = applycform(gfrgb,cform);
% Convert the smoothed image from the sRGB color space to the CIE Lab color space.
l = double(lab(:,:,1)); lm = mean(mean(l));
a = double(lab(:,:,2)); am = mean(mean(a));
b = double(lab(:,:,3)); bm = mean(mean(b));
% Calculate saliency map using color deviation
sm = (l-lm).^2 + (a-am).^2 + (b-bm).^2;
%imshow(sm,[]);
```

```
% Load the image
img = imread('C:\Users\lenovo\Downloads\img.jpg');  % Replace with the path to your image

% Perform saliency detection
saliency_map = saliency_detection(img);

% Display the original image and the saliency map
figure;
subplot(1, 2, 1);
imshow(img);
title('Original Image');

subplot(1, 2, 2);
imshow(saliency_map, []);
title('Saliency Map');
```

Output:



Final Image Enhancement code:

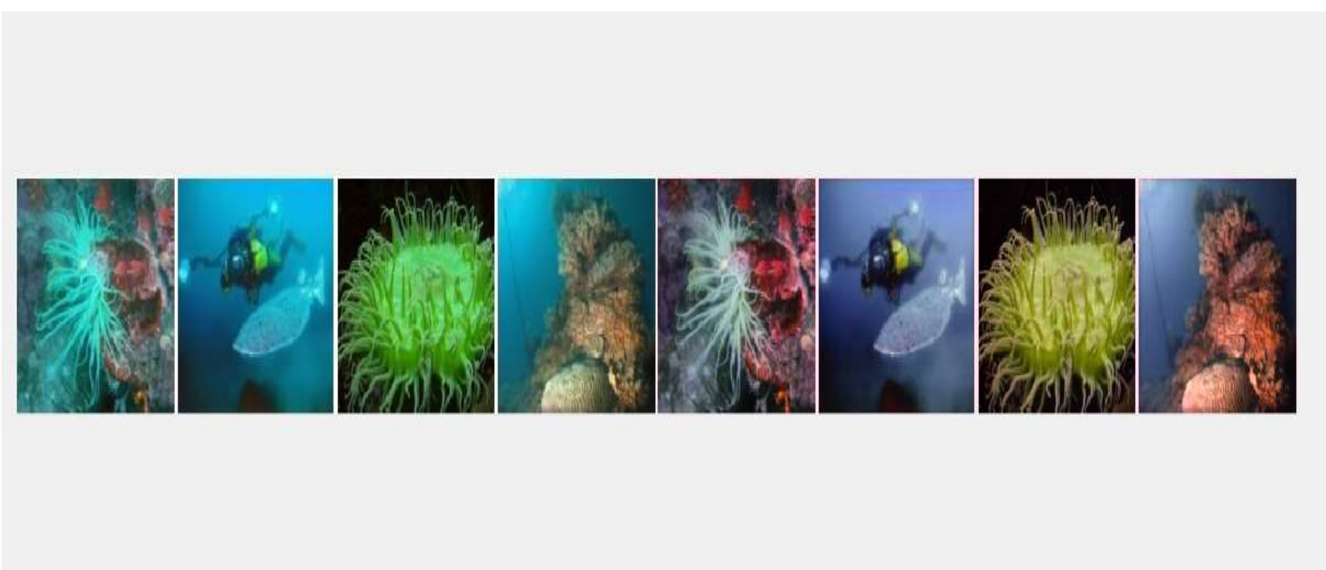
```
close all;
clear all;
clc;

%%% Underwater White Balance %%%

%% Load the image and split channels.

rgbImage=double(imread('C:\Users\lenovo\Downloads\img.jpg'))/255; % Load the image
and convert it to double format, scaling to the range [0, 1]

grayImage = rgb2gray(rgbImage); % Convert the RGB image to grayscale

Ir = rgbImage(:,:,1); % Extract the individual color channels (Red, Green, Blue) from
the RGB image
Ig = rgbImage(:,:,2);
Ib = rgbImage(:,:,3);

Ir_mean = mean(Ir, 'all'); % Calculate the mean intensity for each color channel
Ig_mean = mean(Ig, 'all');
Ib_mean = mean(Ib, 'all');

%% Color compensation
alpha = 0.1; % Define a compensation factor (alpha) for color channels
Irc = Ir + alpha*(Ig_mean - Ir_mean); % Apply compensation to the Red channel (Irc)
alpha = 0; % 0 does not compensates blue channel.

Ibc = Ib + alpha*(Ig_mean - Ib_mean);

%% White Balance
% Concatenate the compensated channels to form a color-corrected image (I)
I = cat(3, Irc, Ig, Ibc);
I_lin = rgb2lin(I); % Convert the color-corrected image to linear RGB space
percentiles = 5; % Define a percentage for calculating image illuminant
```

```
illuminant = illumgray(I_lin,percentiles); % Estimate the illuminant using the linear
RGB image
I_lin = chromadapt(I_lin,illuminant,'ColorSpace','linear-rgb'); % Apply chromatic
adaptation to correct color imbalances
Iwb = lin2rgb(I_lin); % Convert the linear RGB image back to RGB space (Iwb - White
Balanced Image)


%figure('name', 'Underwater White Balance');
%imshow([rgbImage, I, Iwb])

%%% Multi-Scale fusion.

%% Gamma Correction
Igamma = imadjust(Iwb,[],[],2); % applies gamma correction to the white-balanced
image, each pixel raide to power 2

%figure('name', 'Gamma Correction');
%imshow([Iwb, Igamma])


%% image sharpening
sigma = 20; % Set the standard deviation for the Gaussian filter
Igauss = Iwb; % Initialize the blurred image as the white-balanced image
N = 30; % Number of iterations for the sharpening process
for iter=1: N
   Igauss =  imgaussfilt(Igauss,sigma); % Apply Gaussian filtering to the blurred
image
   Igauss = min(Iwb, Igauss); % Ensure the sharpened image does not exceed the
original intensity
end

gain = 1; % Set the gain for the sharpening process
Norm = (Iwb-gain*Igauss); % Calculate the normalized difference between the white-
balanced image and the blurred image
% Apply histogram equalization to enhance contrast in each color channel separately
for n = 1:3
   Norm(:,:,n) = histeq(Norm(:,:,n));
end
Isharp = (Iwb + Norm)/2;  % Combine the normalized difference and the original image
to obtain the sharpened image

% figure('name', 'image sharpening');
% imshow([Iwb,Igauss,Norm, Isharp])


%% weights calculation

% Lapacian contrast weight % Convert the sharpened and gamma-corrected images to
CIELAB color space
Isharp_lab = rgb2lab(Isharp);
Igamma_lab = rgb2lab(Igamma);

% input1
```

```
R1 = double(Isharp_lab(:, :, 1)) / 255; % Extract the L* channel from the CIELAB
representation and normalize it
% calculate laplacian contrast weight
WC1 = sqrt((((Isharp(:,:,1)) - (R1)).^2 + ...
            ((Isharp(:,:,2)) - (R1)).^2 + ...
            ((Isharp(:,:,3)) - (R1)).^2) / 3);
% calculate the saliency weight
WS1 = saliency_detection(Isharp);
WS1 = WS1/max(WS1,[],'all');
% calculate the saturation weight

WSAT1 = sqrt(1/3*((Isharp(:,:,1)-R1).^2+(Isharp(:,:,2)-R1).^2+(Isharp(:,:,3)-
R1).^2)); % Calculate the saturation weight for input1


%figure('name', 'Image 1 weights');
%imshow([WC1 , WS1, WSAT1]);


% input2
R2 = double(Igamma_lab(:, :, 1)) / 255;
% calculate laplacian contrast weight
WC2 = sqrt((((Igamma(:,:,1)) - (R2)).^2 + ...
            ((Igamma(:,:,2)) - (R2)).^2 + ...
            ((Igamma(:,:,3)) - (R2)).^2) / 3);
% calculate the saliency weight
WS2 = saliency_detection(Igamma);
WS2 = WS2/max(WS2,[],'all');

% calculate the saturation weight
WSAT2 = sqrt(1/3*((Igamma(:,:,1)-R1).^2+(Igamma(:,:,2)-R1).^2+(Igamma(:,:,3)-
R1).^2));

%figure('name', 'Image 2 weights');
%imshow([WC2 , WS2, WSAT2]);

% calculate the normalized weight
W1 = (WC1 + WS1 + WSAT1+0.1) ./ ...
     (WC1 + WS1 + WSAT1 + WC2 + WS2 + WSAT2+0.2);
W2 = (WC2 + WS2 + WSAT2+0.1) ./ ...
     (WC1 + WS1 + WSAT1 + WC2 + WS2 + WSAT2+0.2);


%% Naive fusion
R = W1.*Isharp+W2.*Igamma; %to combine pixel values from different images.
%figure('name', 'Naive Fusion');
%imshow([I, Iwb, Isharp, Igamma, R]);


%% Multi scale fusion.
 img1 = Isharp;
 img2 = Igamma;

% calculate the gaussian pyramid
level = 10;
```

```
Weight1 = gaussian_pyramid(W1, level);
Weight2 = gaussian_pyramid(W2, level);

% calculate the laplacian pyramid
% input1
R1 = laplacian_pyramid(Isharp(:, :, 1), level);
G1 = laplacian_pyramid(Isharp(:, :, 2), level);    % to improve images particularly
useful when images appear blurry or lack sharpness.
B1 = laplacian_pyramid(Isharp(:, :, 3), level);
% input2
R2 = laplacian_pyramid(Igamma(:, :, 1), level);
G2 = laplacian_pyramid(Igamma(:, :, 2), level);  % helps improve the overall
brightness and contrast of an image
B2 = laplacian_pyramid(Igamma(:, :, 3), level);

% fusion
for k = 1 : level
   Rr{k} = Weight1{k} .* R1{k} + Weight2{k} .* R2{k};
   Rg{k} = Weight1{k} .* G1{k} + Weight2{k} .* G2{k};
   Rb{k} = Weight1{k} .* B1{k} + Weight2{k} .* B2{k};
end

% reconstruct & output
R = pyramid_reconstruct(Rr);
G = pyramid_reconstruct(Rg);
B = pyramid_reconstruct(Rb);
fusion = cat(3, R, G, B);

figure('name', 'Multi scale fusion'); % applying fusion at each scale, and then
reconstructing the final fused image.
imshow([I, fusion])
```

Output:



Conclusion:

In summary, Our Image enhancement method employs a single-image approach without specialized hardware. Utilizing Gaussian and Laplacian pyramids, it tackles scattering and absorption challenges, delivering improved exposure, global contrast and sharp edges. Integrating white balancing, gamma correction, sharpening and multi-scale fusion, it enhances image quality significantly.