# Homework 1 - Computer Vision, 2018 Spring
# Team35

Teammates:0416088 鄭甯遠,0650249 劉岑陽子,0660819 司菲 Siffi Singh

For the given dataset, following tasks are asked to complete:
1. Hybrid Image (MATLAB)
2. Image Pyramid (Python)
3. Colorizing the Russian Empire (MATLAB)

In this report, we will be explaining each part in the following way:
1. Steps to run the file.
2. Explanation of Code Logic
3. Results
4. Inferences from Result

## I. Hybrid Image

A hybrid image is the sum of a low-pass filtered version of the one image and a high-pass filtered version of a second image. There is a free parameter, which can be tuned for each image pair, which controls how much high frequency to remove from the first image and how much low frequency to leave in the second image. This is called the "cutoff frequency".

### 1. Steps to run file

- There are three files in MATLAB for the hybrid image part. The first file is the driver file i.e. **main.m.**
- The other two files are function being called from the main function, namely **hybrid.m** and **filter_image.m.**
- To run the file, simply put the image 1 and image 2 in the same folder as the files or specify the path of the two images.
- Run main.m, in the output you will get low_frequency of Image 1 and high_frequency of Image 2.
- The final output is the hybrid image of Image 1 and Image 2.

### 2. Explanation of Code Logic

These are the three functions that have been used:
- main.m
- Filter_image.m

- Hybrid.m

Key Point:
% Combine the high frequencies and low frequencies
hybrid_image = low_frequencies + high_frequencies;

The key functions of main.m is to:

1. Read data values from the file.
2. Call filter_image.m function to get low_frequency and high_frequency versions of Image 1 and Image 2 respectively, based upon the cutoff frequency.
3. Call hybrid.m to visualize the hybrid of the two images obtained from addition the low_frequency and high_frequency image from filter_image.m.
4. Display results for low_frequency and high_frequency and the hybrid image.

The key functions of filter_image.m is to:

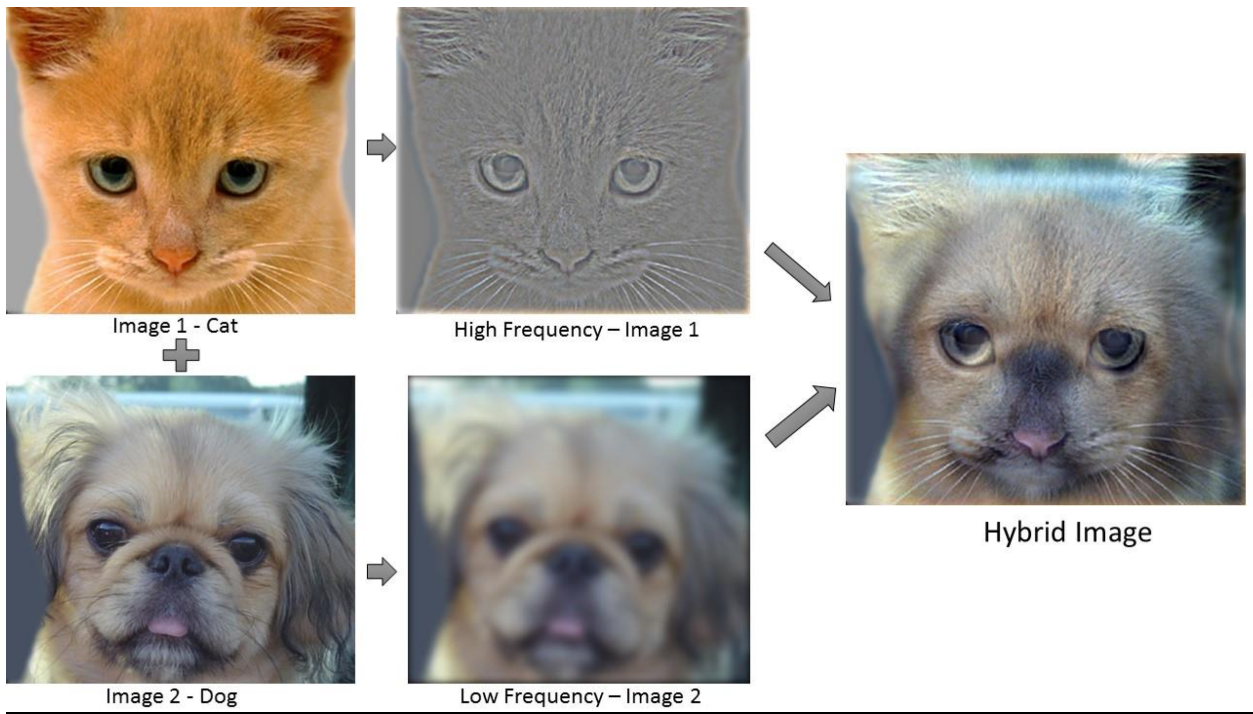
1. This function takes two arguments, the image to be filtered and the gaussian filter with parameters using hsize as the cutoff_frequency*4+1, and the cutoff_frequency as the sigma for gaussian.
2. To create the low_frequency version, we simply pass image 1 and formed filter to the filter_image function, as the fspecial function returns a rotationally symmetric Gaussian low pass filter of size cutoff_frequency*4+1 and standard deviation as the cutoff_frequency.
3. To create the high_frequency version, we subtract the output from filter of image 2 and filter passed from the image 2 for further formation of hybrid image.

The key functions of hybrid.m is to:

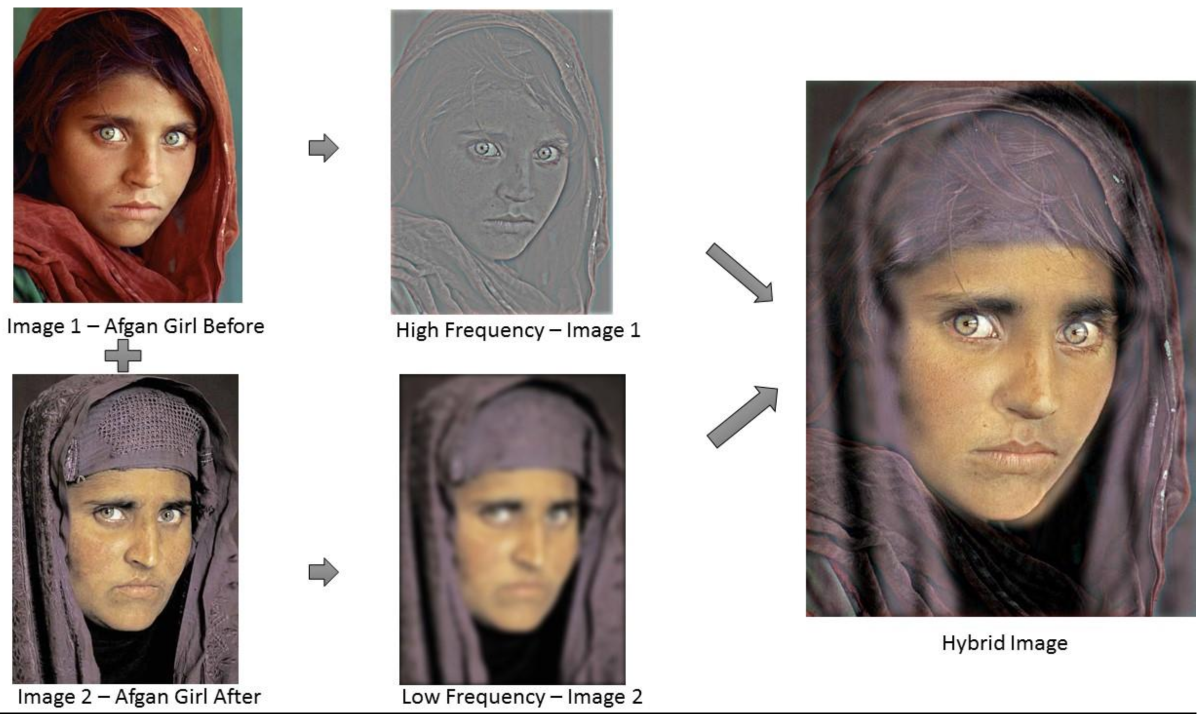
1. Visualize the output, i.e. our final hybrid image.
2. Using hybrid.m we can even visualize output at different scales, and create multiple downsampled versions.

## 3. Results

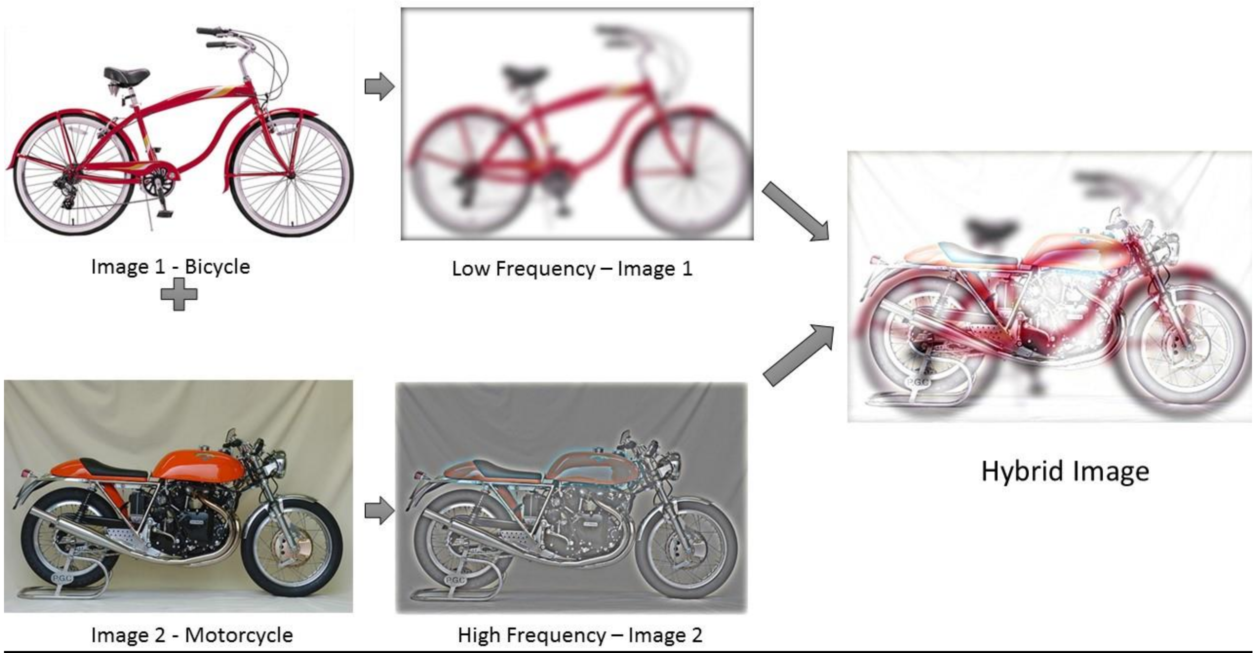
- Following is the output from the input dataset given in hw1_data file.
- Additionally, the results for a few images other than the ones provided in the dataset have also been executed and their results have been included too.



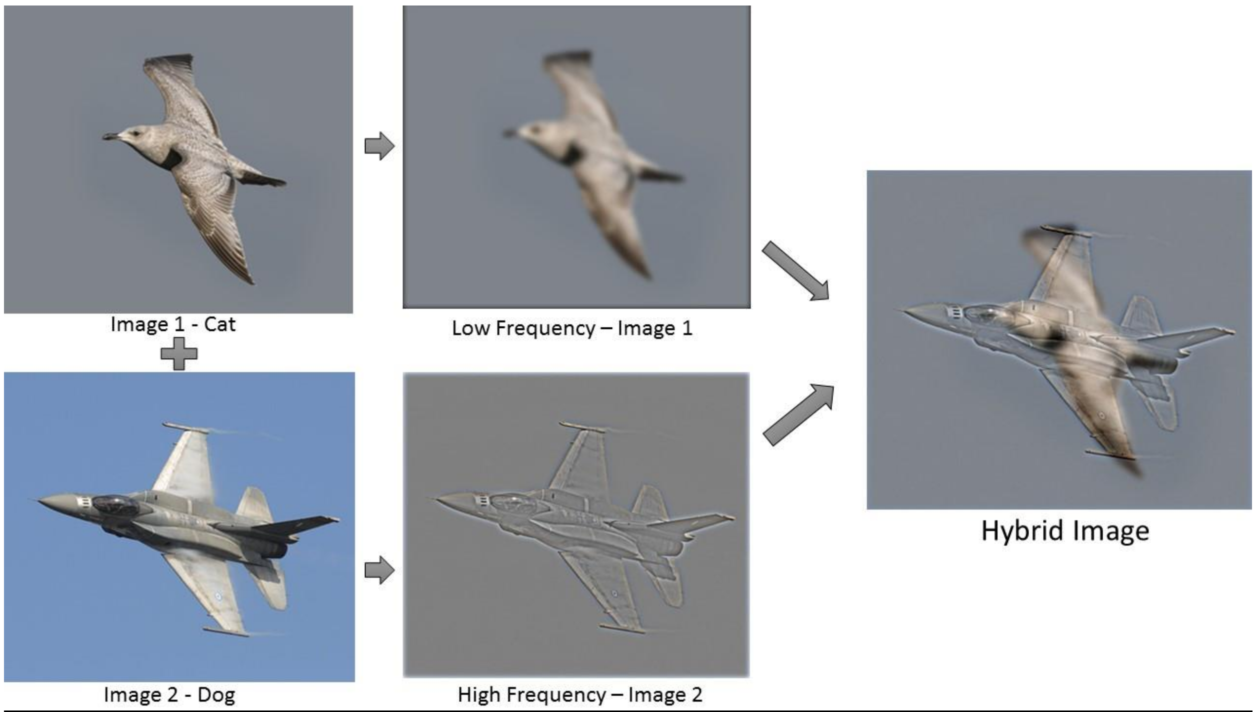
**Cutoff_Frequency = 5**



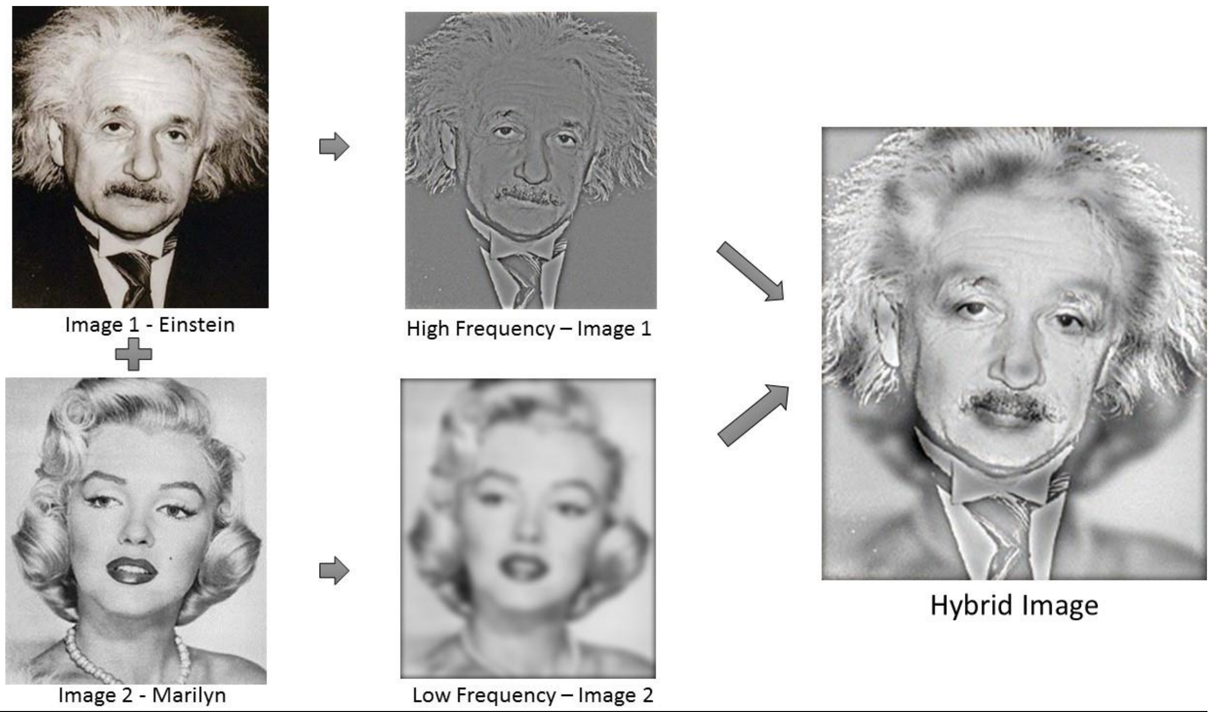
**Cutoff_Frequency = 5**



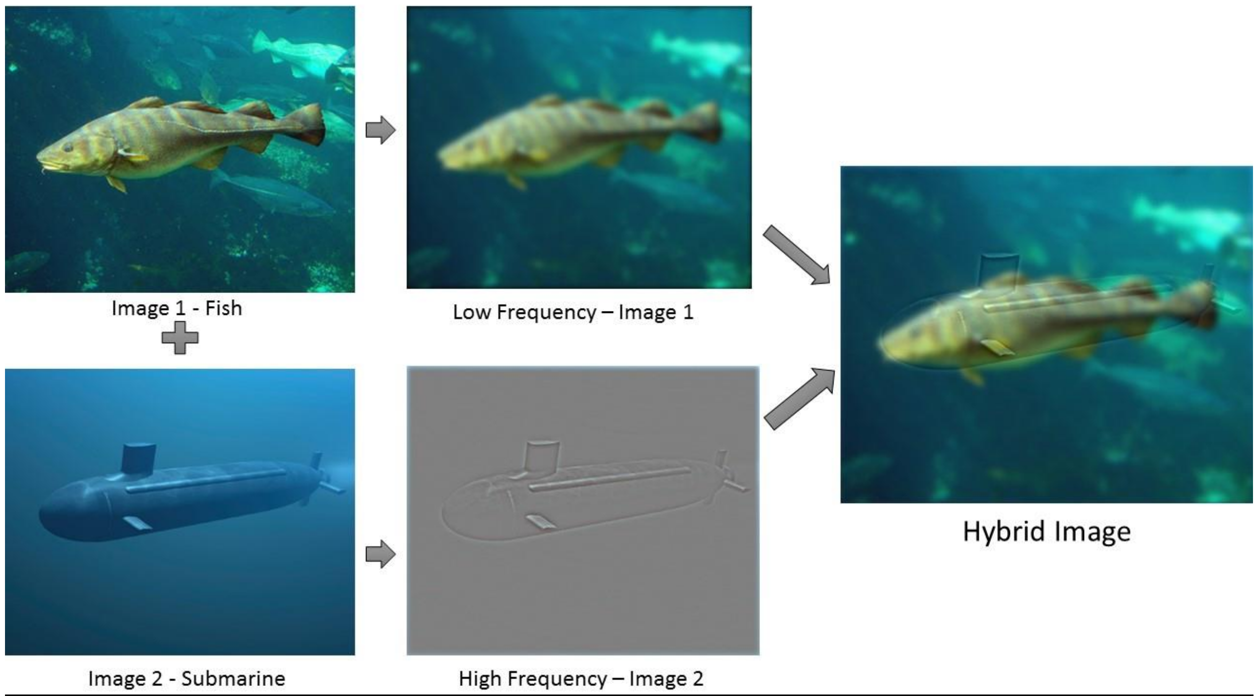
**Cutoff_Frequency = 5**



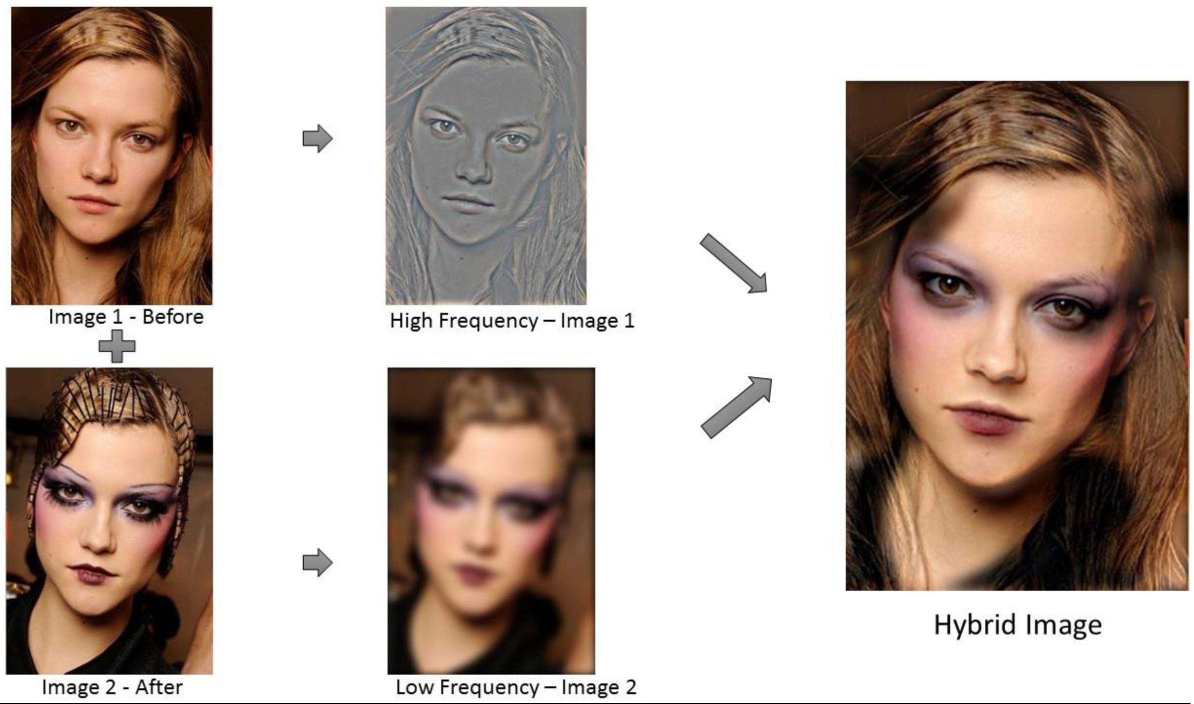
**Cutoff_Frequency = 3**



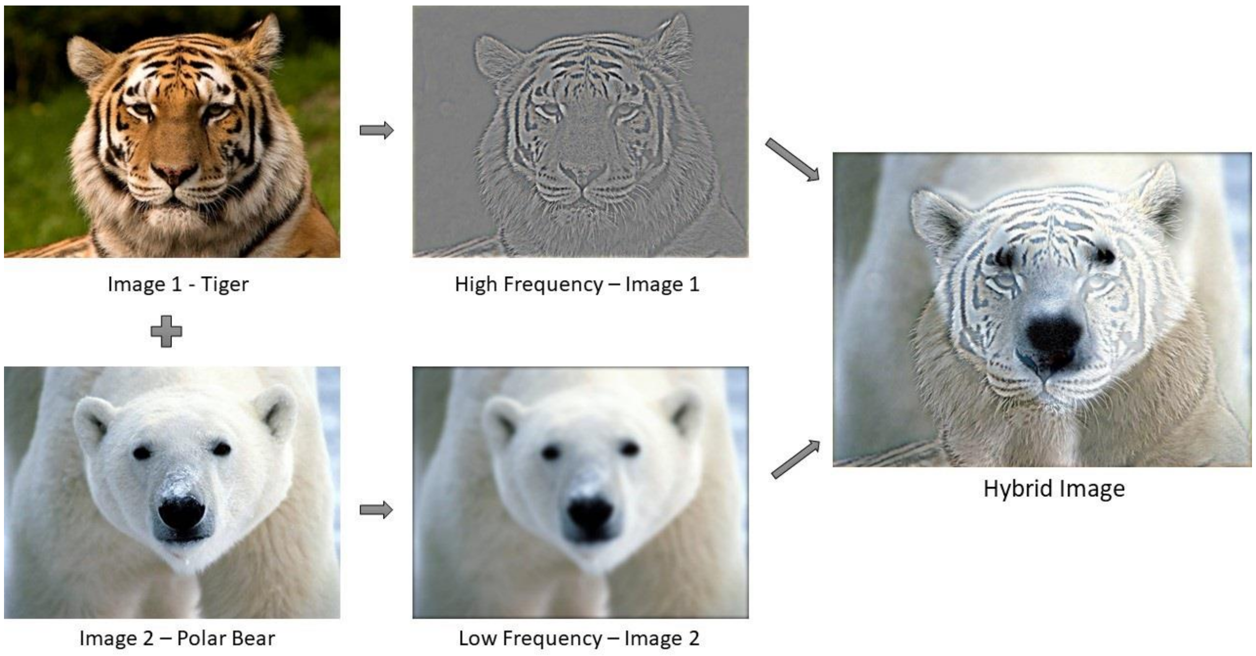
**Cutoff_Frequency = 3**



**Cutoff_Frequency = 3**



**Cutoff_Frequency = 5**



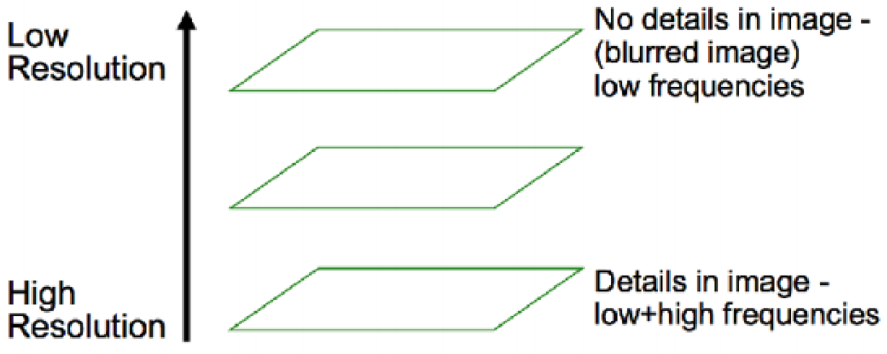
Additionally, another pair of image of tiger and polar bear have also been tested with cutoff frequency = 7.

## <u>4. Inferences from Result</u>

- Distinct cut-off frequencies were chosen for each pair of image, based on trails.
- The size of image being merged should be the same, and the output image will be the same.
- As the cutoff frequency increases, the image, low frequency image in particular, get more blurred.

## II. Image Pyramid

An image Pyramid is a collection of representations of an image.


Low Resolution
High Resolution
No details in image - (blurred image) low frequencies
Details in image - low+high frequencies

A collection of images at different resolutions.

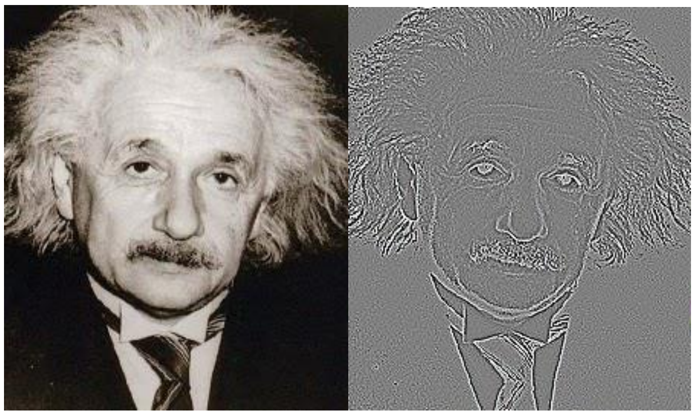
### 1. Steps to run file

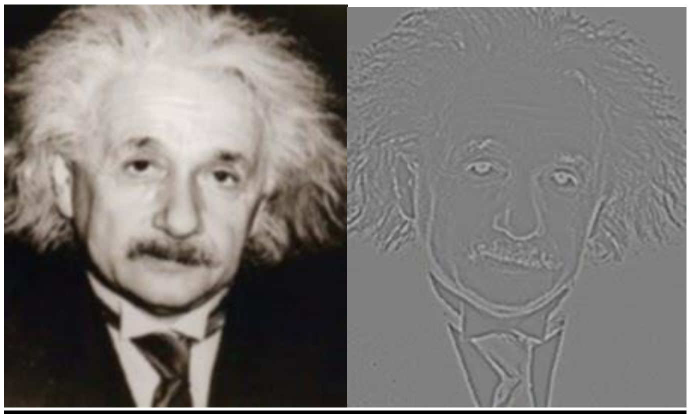
There is only one file called pyramid.py. To use it, simply type "python pyramid.py directoryToTheFile. The output will be put inside in the output folder. If there is no folder named output in working directory, please create one yourself.

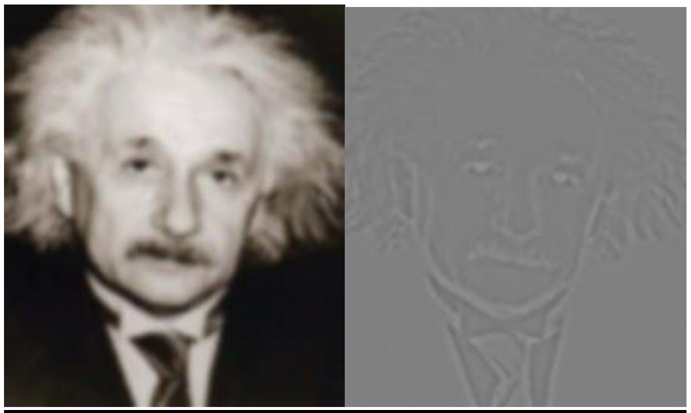
### 2. Explanation of Code Logic

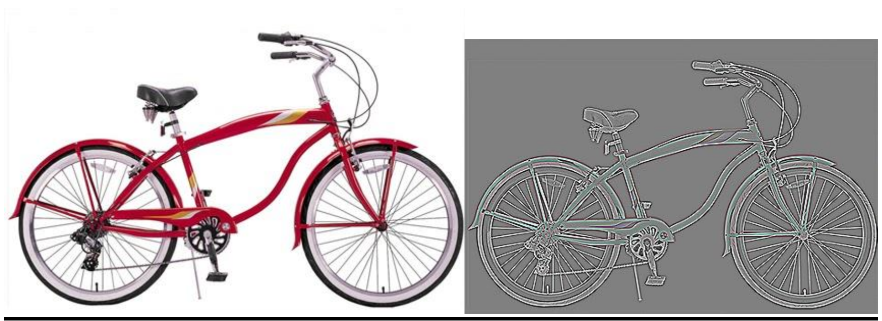
The code logic is relatively easy. First of all, I apply gaussian filter to the photo. Secondly, LaPlacian filter and then add 128 to the result.

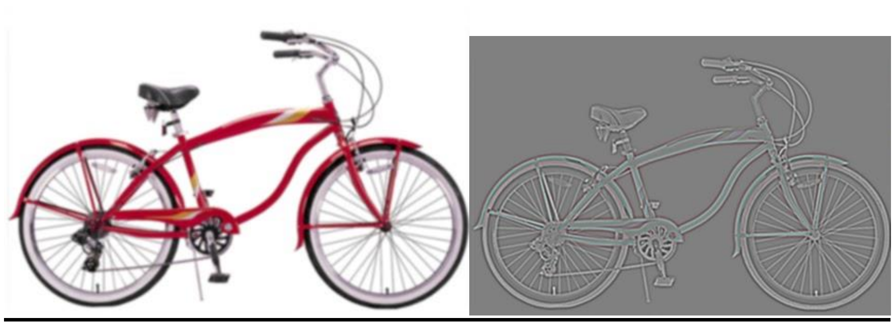
### 3. Results

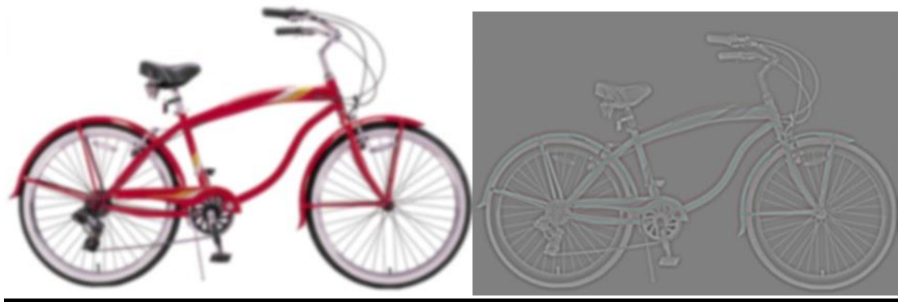








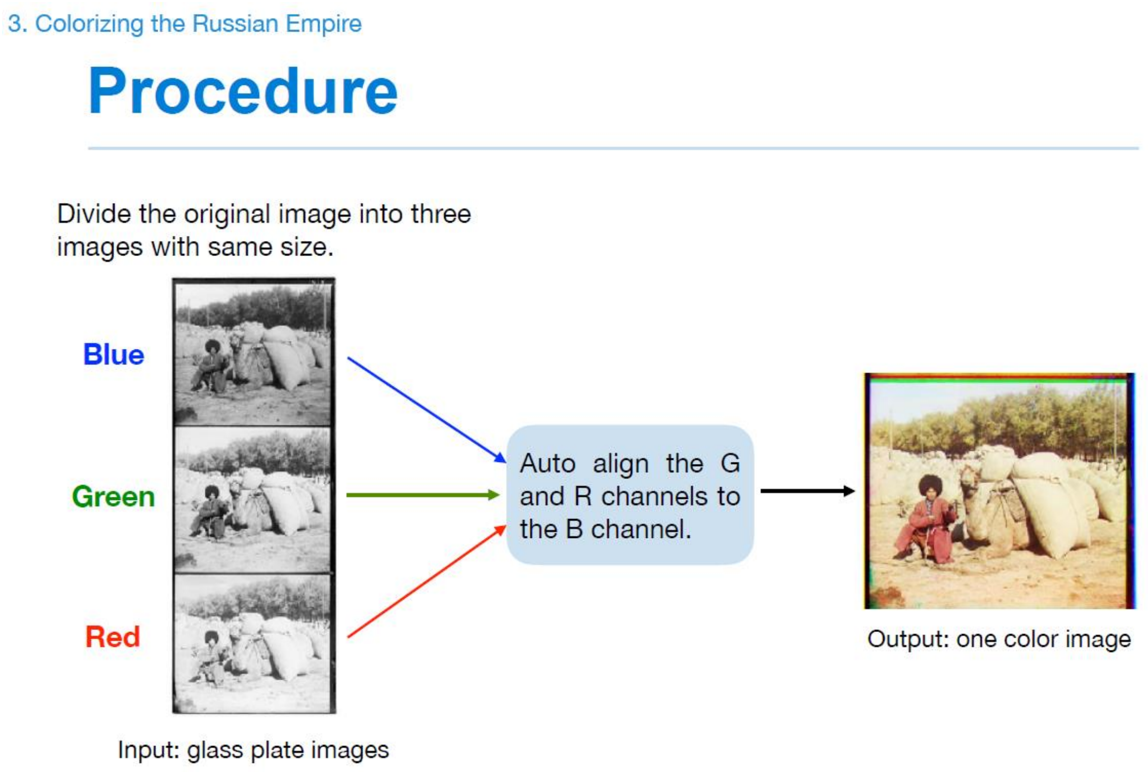




**4. Inferences from Result**

The result of LaPlacian transform is highly affected by Gaussian filter. By applying a little bit more Gaussian filter to the image, one loses details/edges in LaPlacian transform quickly.

## III. Colorizing the Russian Empire


3. Colorizing the Russian Empire
Procedure
Divide the original image into three images with same size.
Blue
Green
Red
Auto align the G and R channels to the B channel.
Output: one color image
Input: glass plate images

**1. Steps to run file**

There are two files for this task. One is for smaller size images while the other file is for larger images. Changing the input name and run those files in Matlab.

Smaller size: 3xx*1xxx

Larger size: 3xxx*9xxx

**2. Explanation of Code Logic**

**Preprocess:**

Splitting 3 channel is necessary. And then cutting excessive white and black borders might help aligning different channels. Using edge detection to find the black edges and crop the images. Last step is zero padding. Because we want to shift image and calculate difference square sum, a larger images is necessary in order to keep window size the same as each channel.

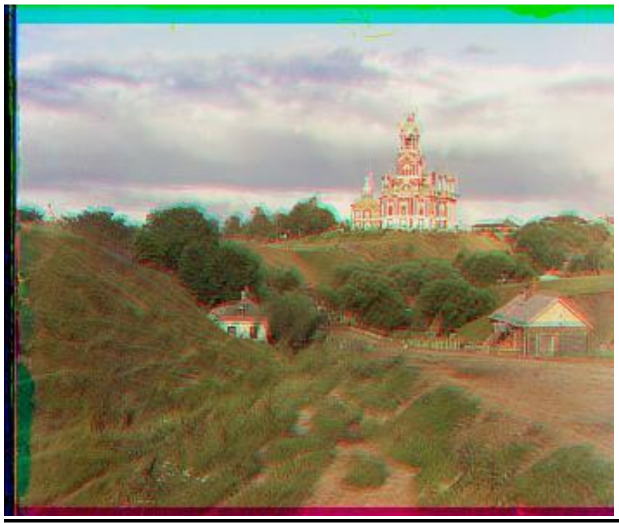
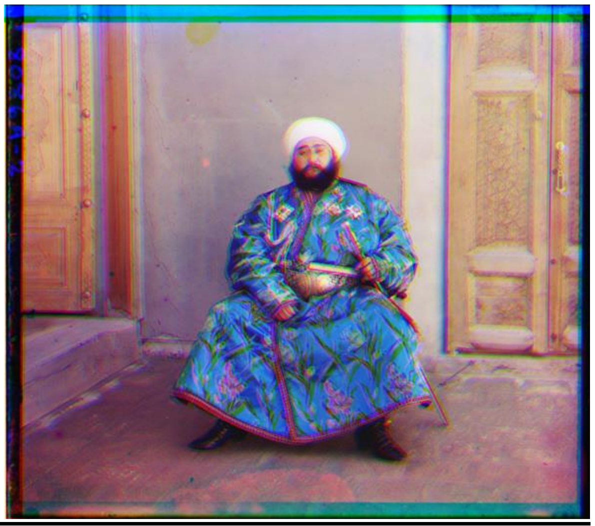
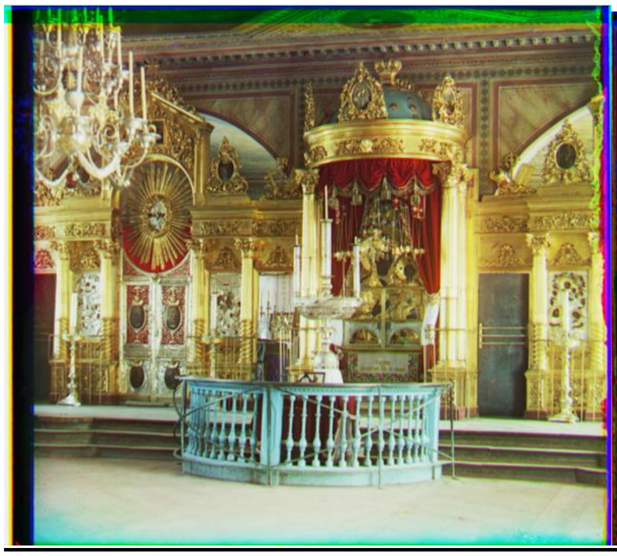
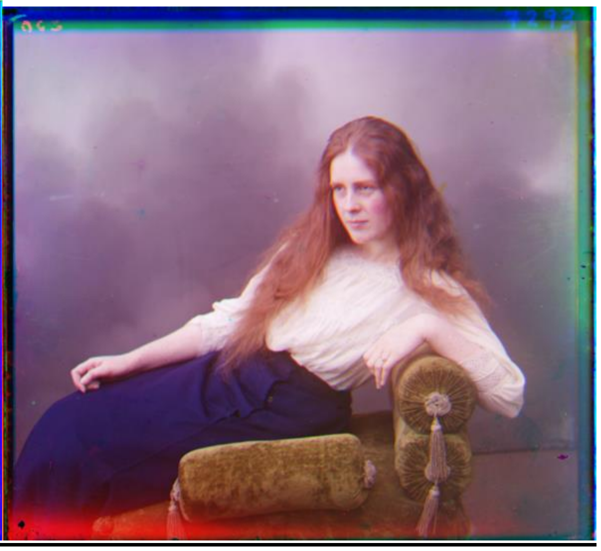
A. **For smaller images**

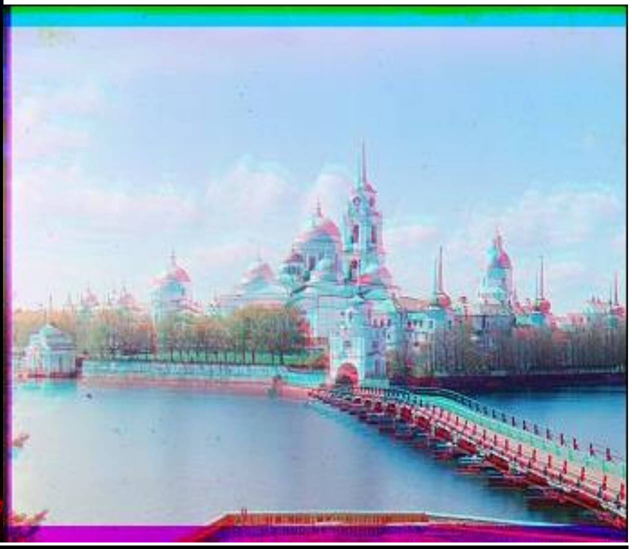
I compared the intensity difference between two channels. The main idea behind this is that in different channel, the intensity should be similar. Minimum difference square sum is used to compare the intensity. With different row and column offset, we can get different square sum value. Among them, we can select the optimal offset which has the minimum difference square sum.

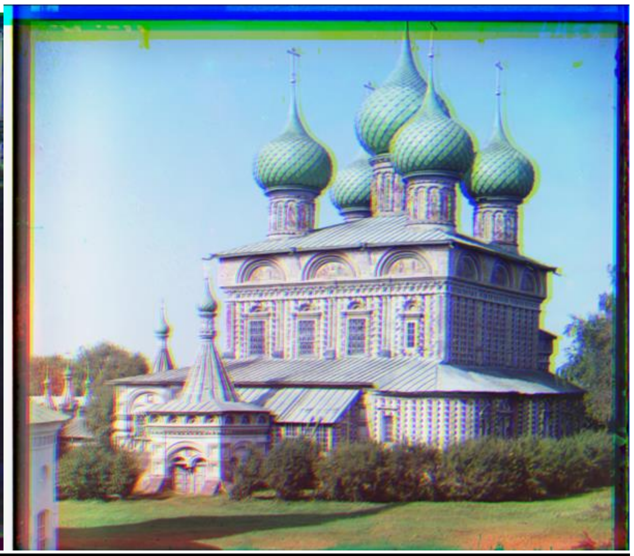
B. **For larger images**

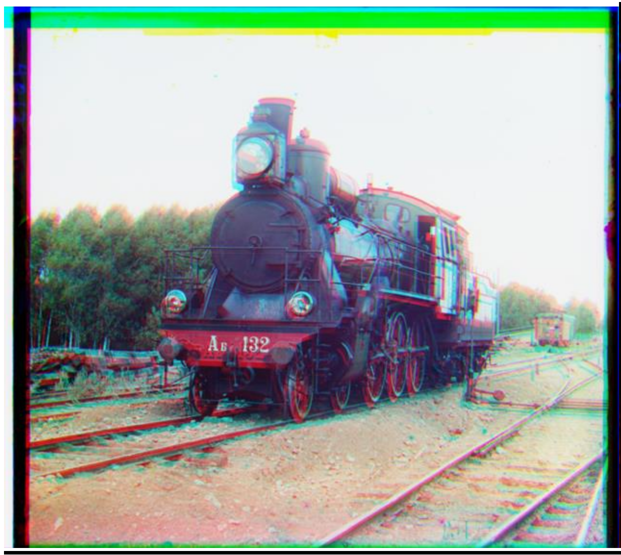
Because it takes long time to do the exhaustive search when the input is large size. Therefore, I tried to resize the input image into smaller size and do the search.Which reduces computation time. After optimal offset is chosen, resizing each channel back to the original size. And then put three channels into R,G,B.

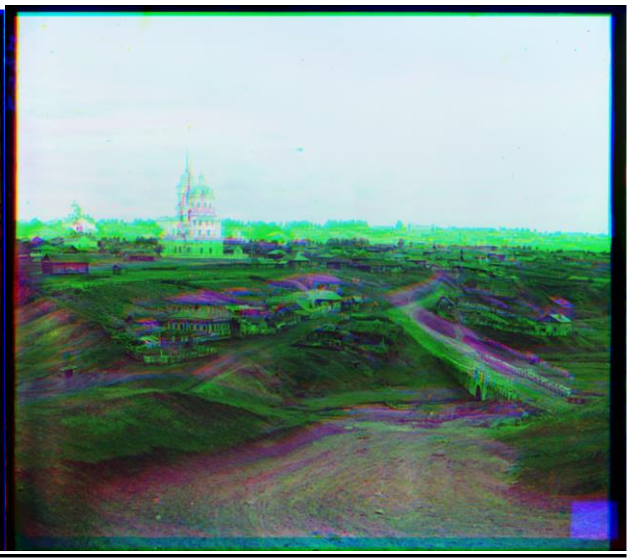
## 3. Results

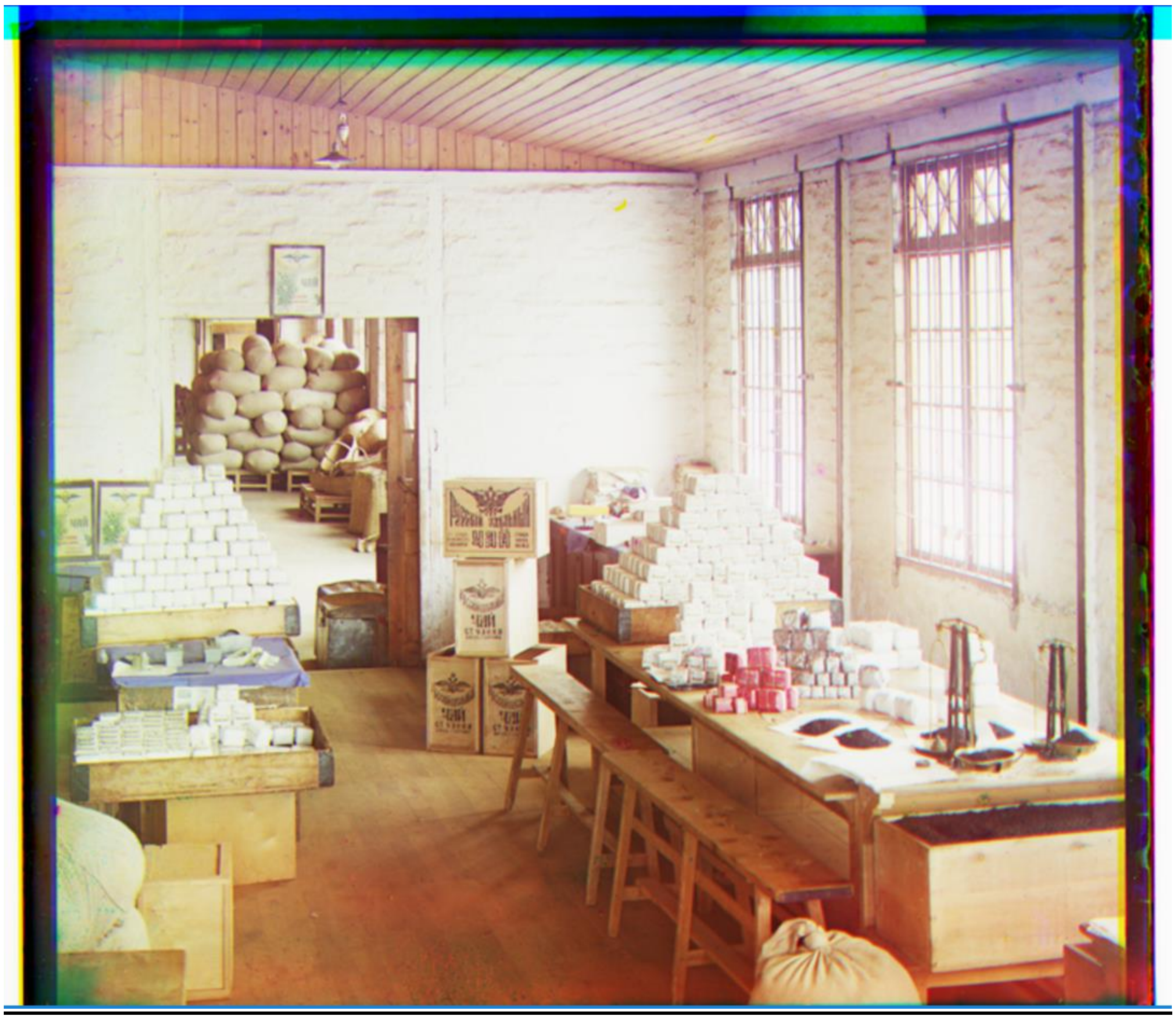















## 4. Inferences from Result

A. Normalization image:
   I tried to normalize each channel but found that normalization makes no difference. The results are the same with/without normalization before doing the search.
B. Choosing different search range can produce different results. A general range is chosen to fit most images. However, there are some poor images while some are good.