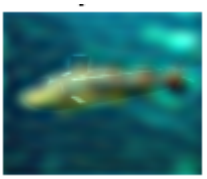
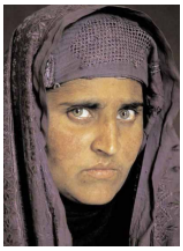
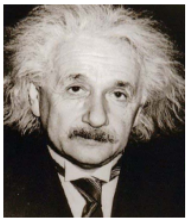
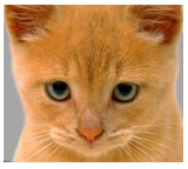
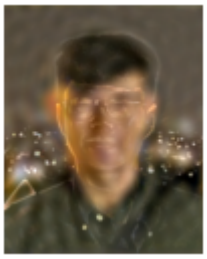
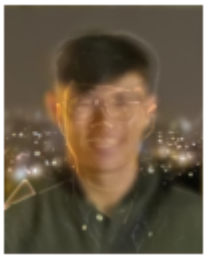
Computer Vision - HW2

Task 1: Hybride Image

1. **Introduction**
   A hybrid image, an image that appears to change its interpretation depending on distance, is implemented by combining a low-pass-filtered image and another high-pass-filtered image. From the distance we can see the whole picture(low frequency part), while from standing closely you will see the detail edges(high frequency part). We use Ideal and Gaussian filters to implement this task.
2. **Implementation procedure**
   Step1: Use Fourier Transform to get the image Spectrum
   First, we use Fourier Transform function(fft) from numpy. Second, shifting the zero frequency component to the center by using fftshift(). Eventually, using take the absolute value and log the make the value between 0 and 255 to visualize them in an image.
   Step2: Implement Ideal low-pass and high-pass filter
   We set the cut-off frequency to make a circle of cut boundary, which we found the shortest image boundary and divided it to get the radius of the circle. To make the pixel position into the center of the circle, we shifted it by minusing the offset. Next, we follow the formula of ideal low-pass and high-pass filter H(u,v) to determine 0 or 1 value by cut-off frequency. Finally, do the inverse Fourier Transform to get the RGB value.
   Step3: Implement Gaussian low-pass and high-pass filter
   Identically, we get the cut-off frequency in the same way as the ideal filter, except we change the formula to the Gaussian filter form, which is using Gaussian distribution to ideal the pixel distribution.
   Step4: combining the low-pass and high-pass images
   After filtering the images, we combine the low and high frequency image together to obtain the hybrid image we.
3. **Experiment Result**

Group 5

original	Ideal	Gaussian	(D0)
 			8
 			8
 			12
 			10
 			8
 			8

 				**12**

The cut-off frequency D0 is computed by shorter boundary of image divided by 2 and multiply the cut-off ratio which stands for the formula D0= min(img.shape) / 2 * cut-off ratio .

4. Discussion

4.1 As implementing the ideal and gaussian filters and hybrid the images functions, we found out that we should set the outputs ,which are the filtered images and filtered spectrums, into float type or we might encounter an overflow issue which images will look like below.


4.2 To choose a good cut-off ratio, we have tried many different values to see the result. We might need to find a good ratio that will not make the image too blurry after filtered by a high-pass filter. Meanwhile, the image edges should also be clear after filtered by a low-pass filter.

4.3 Different image sizes might cause some computation errors or some aligning issues. We solved it by finding the minimum row or column to compute our cut-off frequency.

4.4 The Fourier transform function(fft) in numpy only supports gray and one channel in particular. Therefore, we converted R, G, B channels into gray images and do fft respectively.

5. Conclusion

During the experiment, we realized that the Gaussian filtered results are smoother than the ideal ones. In addition, the cut-off frequency plays a role in how well the hybrid images display. In

conclusion, we have learned how to produce hybrid images by converting images to spectrums by Fourier Transform and filtering the frequency information with filters to generate blurry and contoured effects.

Task 2: Image Pyramid

1. Introduction

An image Pyramid is a collection of representations of an image from its high resolution to its low resolution(blurred image).We practice how to build Gaussian pyramids, Laplacian pyramids, and show the result on frequency domain.

2. Implementation procedure

Step 1: Use Gaussian Filter to Smooth images

Before downsampling each image to a smaller one, we use the Gaussian filter to smooth the image to remove high frequency components that could cause aliasing.

The kernel is size 3*3, mean equal to zero, and standard deviation equals one. The filter follows f(x, y) = $\frac{1}{2\pi} e - \frac{(x^2+y^{2)}}{2}$, and normalize the kernel by dividing with the sum of all elements of the kermel. We then use the fixed kernel to do convolution with images (called “Conv2D” in source code).

Each image is padded values on the four sides of the image, before the convolution procedure. Our padding strategy is that every padding value is the closest pixel’s value on the image; in short, we pad the same values of the edge on each side. Then, follow the formula to finish the convolution.

$G[i, j] = \Sigma_{u=-k}^{k} \Sigma_{v=-k}^{k} H[u, v] F[i - u, j - v]$

$G[i, j]$ is the output pixel of the convolution [i, j]. $H$is the gaussian kernel. F is the input image. The k equal to 1 since our kernel size 3*3.

Step 2: Down Sampling

The smoothed image is then processed with down sampling procedure. The ratio of down sampling is set as 2, so the image

reduces to ¼ times smaller each time by taking one pixel every two pixels of the image.
Repeat **step.1** and **step.2** 5 times, we are able to get a gaussian pyramid with 5 layers.

Step 3: Up Samping
To do images up samping, we simply let the 4 pixels of the up sampling image equal to the pixel value which is considered as the same relative position on its lower resolution pair.

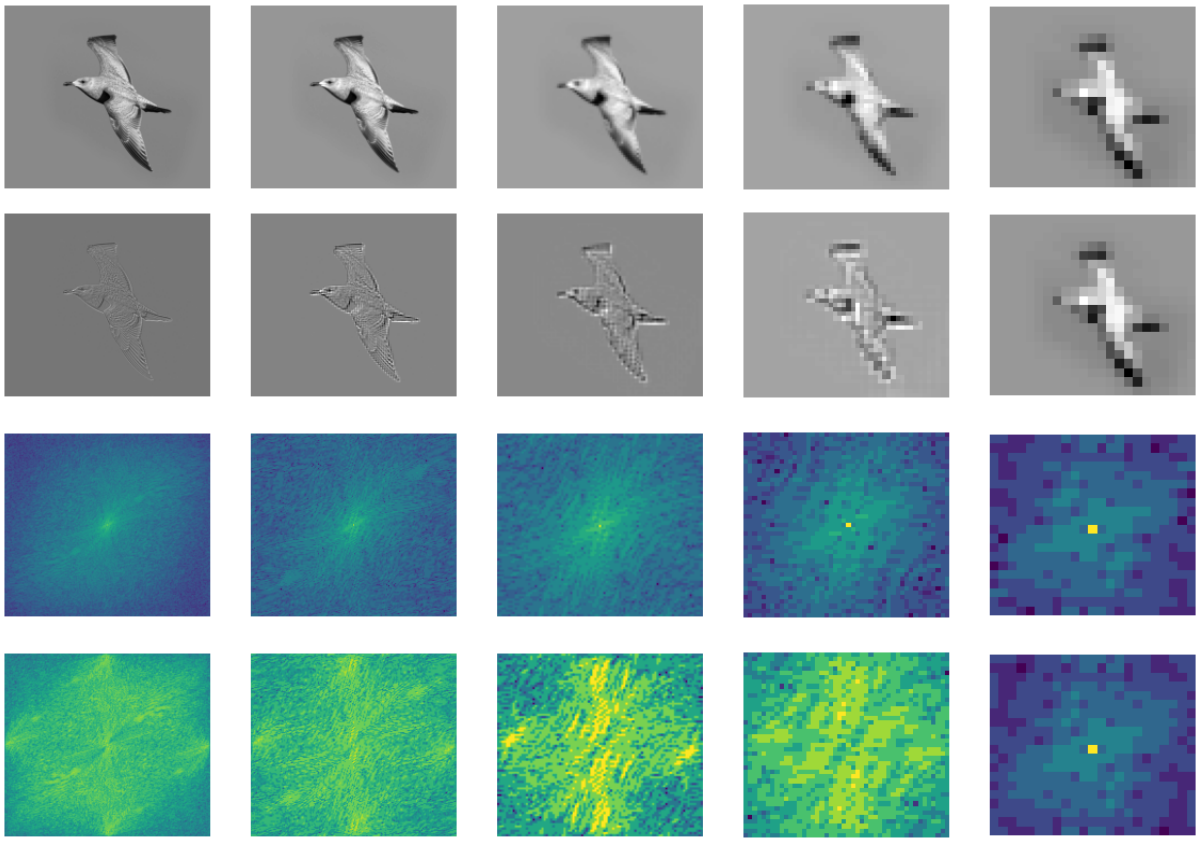
Step 4: Laplacian
A laplacian pyramid is similar to a gaussian pyramid but saves the difference image of the blurred versions between each image level. Thus, the process is that an image of some layer pixel-wisely minus its blur up-sample one.
By using up-sampling images from each layer, we can derive the laplacian pyramid.

Step 5: Magnitude spectrum
To get a magnitude spectrum of pyramids, we use the built-in function numpy.fft.fft2 to do discrete time Fourier transform.

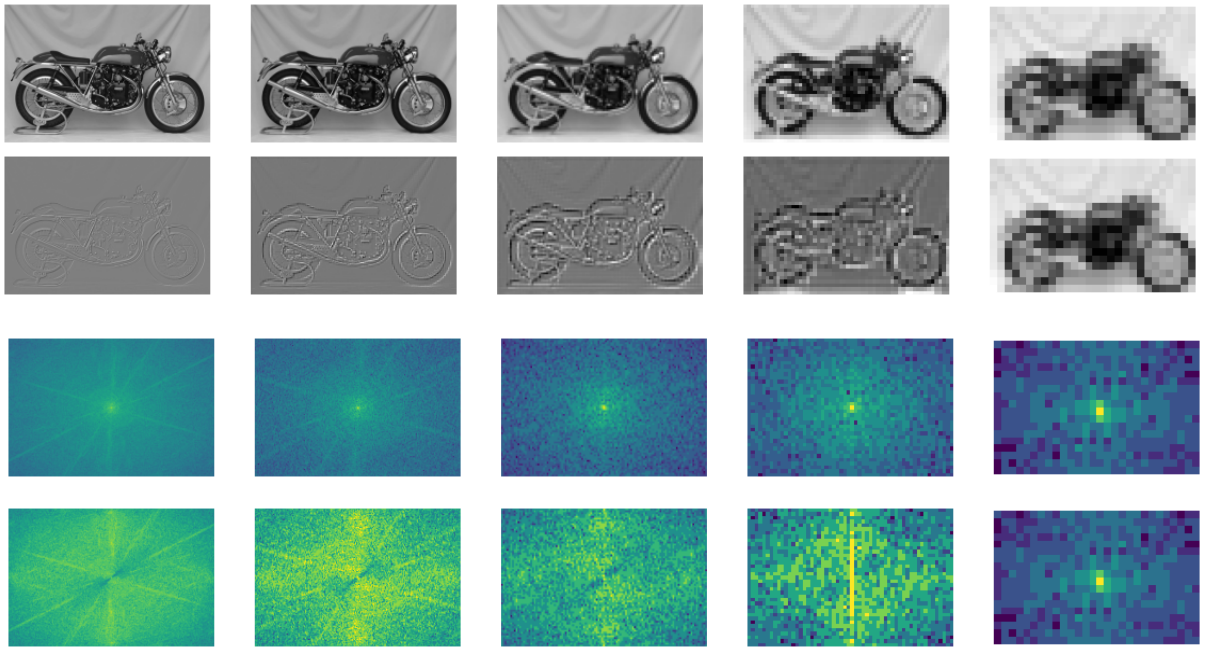
3. Experiment Result

the order of images is gaussian pyramid, laplacian pyramid, magnitude spectrum of gaussian pyramid and laplacian respectively

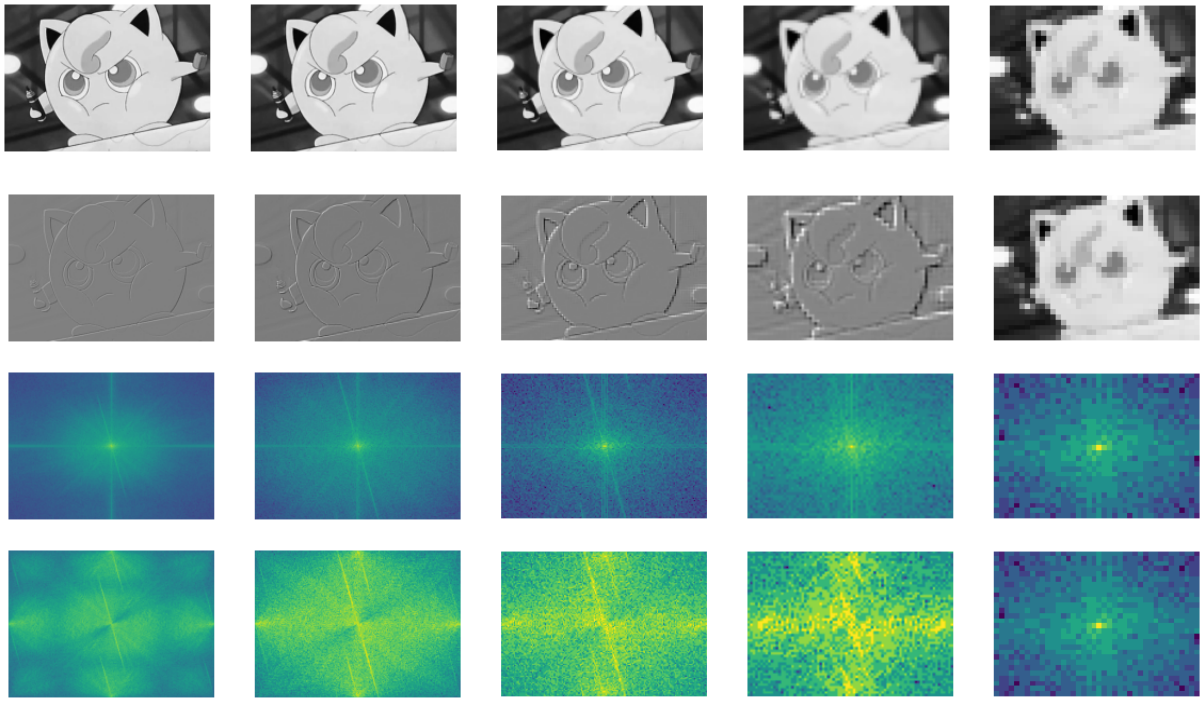
- The result of TA’s data (1)



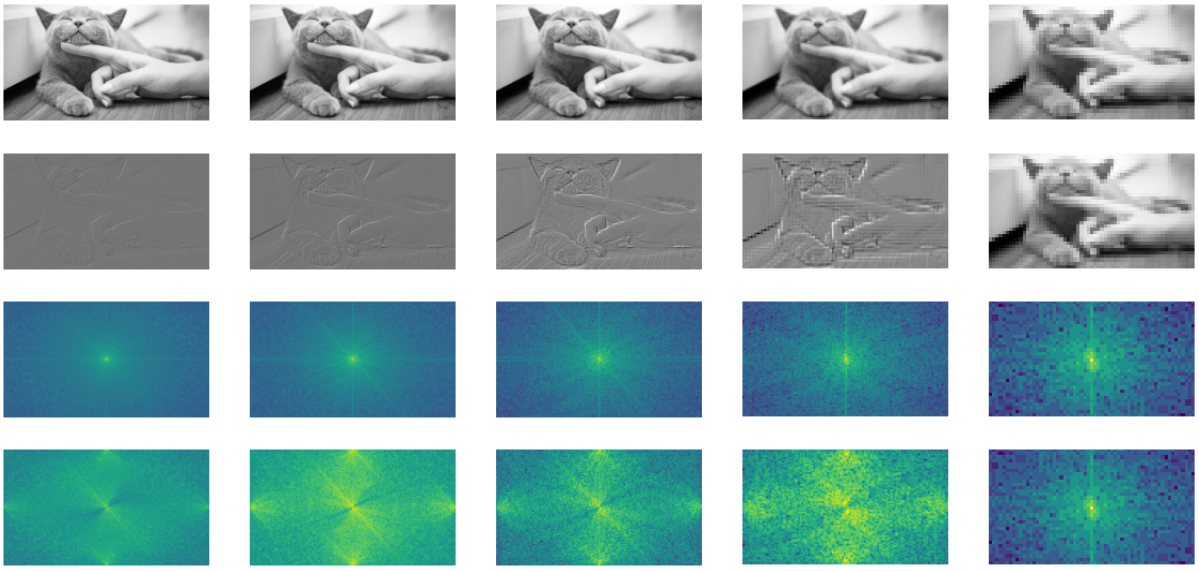
- The result of TA’s data (2)



- The result of our own data (1)



- The result of our own data (2)



4. Discussion

4.1. We notice that different padding strategies cause different effects. At the beginning, we simply use zero padding, however, the output image shows gray lines on the edge of the image due to the zero value, shown below.

As a result, we choose to use the padding method, described in the procedure section to solve this problem. We apply the same color value of the edge to maintain the similar color after convolution.

4.2. We want to observe the effect of smoothing. Before doing down sampling, we skip the convolution step with the gaussian filter. The low resolution image layers look more coarse than the smoothing ones.

It proves that smoothing can effectively prevent aliasing and make the output image better.

4.3. Observing the magnitude spectrums of the gaussian pyramid from the high resolution to low resolution, we find out that the high frequency part is decreasing because the gaussian filter is a low pass filter. On the other hand, the image layers of the laplacian pyramid keep the residual part between the original image and the up-sample image, losing some information, so we can see the some low frequency parts are miss and leaving high frequency information on the spectrum of the laplacian pyramid.

5. Conclusion

Following the procedure above, we are able to build image pyramids. By using a gaussian filter to smooth, it avoids the aliasing problem. Instead of applying zero padding, we use the other strategy to get more complete downsampling images. Both the gaussian pyramid and laplacian pyramid are executed well. According to the results, the magnitude spectrums show that the

gaussian filter removes the high frequency part and the higher level of the gaussian pyramid the more coarse. The laplacian pyramid remains the residual part of the image information.

Task 3:Colorizing the Russian Empire

1. Introduction

The goal is automatically to produce a color image from the digitized Prokudin- Gorskii glass plate images with as few visual artifacts as possible. We learn how to extract the three color channels from the glass plate, and then align one above the other to combine a single RGB color image with a fast and efficient procedure.

2. Implementation procedure

Step 1.
Load the picture and crop the boundary of the picture and get three channels Blue,Green,Red.

Step 2.
Use “Task 2” method “Image pyramid” on three channels respectively .

Step 3.
Use NCC(Normalized Cross Correlation) algorithm to determine two arrays' similarity . Given two image arrays ‘A’ and ‘B’ , we fixed one array (‘A’) and translated another array(‘B’) in a certain range in each pyramid layer . It will get the how much array(‘B’) should translate to get the highest NCC score in greedy search . Finally , we accumulate each layer’s translation value .
(We set the translated distance is 3)

Step 4.
Doing step 3 for (blue,red) and (blue,green) to get translation value for alignment

Step 5.

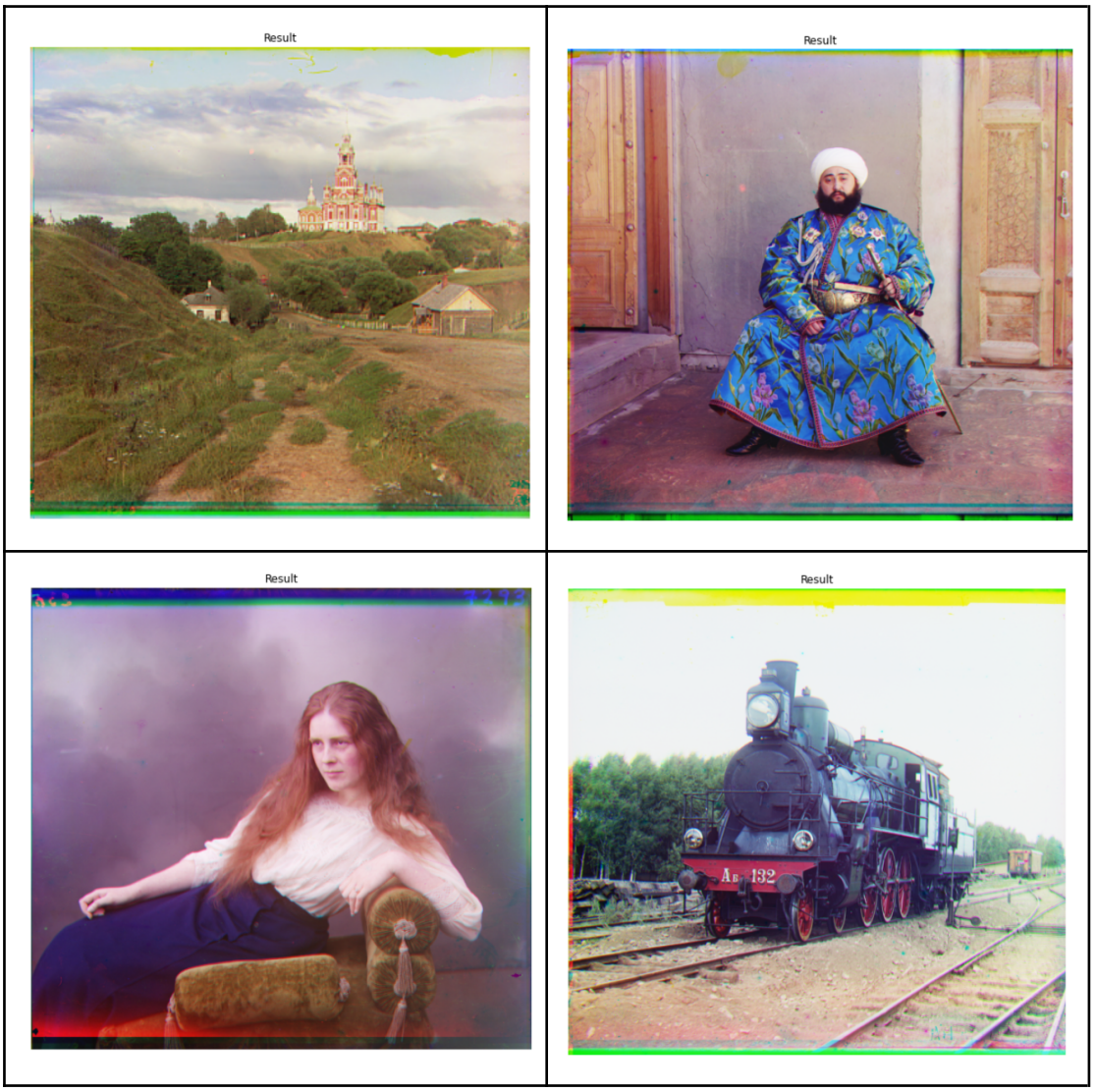
After translation , stack RGB and plot it .

NCC score :

$$NCC_{score}[x,y] = \frac{E[(x-E[x])(y-E[y])}{\sigma_x * \sigma_y}$$

3. Experiment Result

Execution time for an image : 10s (average)
We just show four pictures of the given dataset , other pictures can align too .



4. Discussion

4.1. In this task , we use task 2 method to generate an image pyramid . So the layer in the image pyramid is a parameter that we can control . We observe that if the image pyramid's level is too high , the result is worse . We provide the results below (Figure 1 , 2).

4.2. We observe that when we blur the photo by using task 2 method , the result is not aligned . However , when not blurring the photo and only use downsampling , the result is aligned in the same settings .

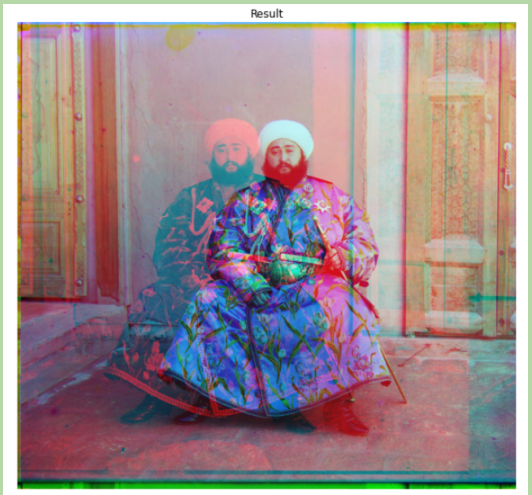
4.3. In the beginning , we used a naive method instead of using an image pyramid . The naive method is using the NCC algorithm only on raw images , and the result is not good . The reason is the receptive field is smaller than using the image pyramid method in the same parameters . If we want a large receptive field in raw image , the computational complexity will also increase and costs a lot of time .

(Figure 1)
Level = 6

(Figure 2)
Level = 7

Group 5


Gaussian Pyramid level 6 Result
Gaussian Pyramid level 5 Result
Gaussian Pyramid level 4 Result
Gaussian Pyramid level 3 Result
Gaussian Pyramid level 2 Result
Gaussian Pyramid level 1 Result



5. Conclusion

We can do the Colorizing task simply by three main steps : remove image border , splitting input image into RGB and do normalized cross-correlation(NCC) . We think that it still has some potential improvement , like the alignment issue .

Work Assignment Plan with team members

We discussed and finished this assignment together.