CV HW2 Report

Group 13

Hybrid Image

Introduction

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

We follow the procedure of homework specification to get filtered version image. After getting the filter version image, we can add them together to generate the hybrid image

Implemntation details

Step 1: Use the Fourier Transform of input image and shift the zero-frequency component to the center => F(u,v)

In fft() function, Use fft2() function to compute Fourier transform and fftshift() function to shift the center

Implemntation details

Step 2: Multiply F(u,v) by filter H(u,v)

In the first, we need to get the lowpass filter. The highpass filter can get by 1 - lowpass filter

D(u,v) is the distance between center and other points in frequency domain

Ideal lowpass filter

$$H(u,v)=\begin{cases}1 & \text{if } D(u,v) \le D_0 \\ 0 & \text{if } D(u,v) > D_0\end{cases}$$

Gaussian lowpass filter

$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

Gaussian highpass filter

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

D_0: cutoff frequency

$D(u,v) = (u^2 + v^2)^{1/2}$

Implemntation details

Step 2: Multiply F(u,v) by filter H(u,v)

We use the D(u,v) and follow the ideal / Gaussian low pass filter formula of the last section to generate the filter H(u,v). After computing the H(u,v), we can multiply F(u,v) by filter H(u,v) to get the filtered version image

Implemntation details

Step 3: Compute inverse Fourier transform, take the real part and compute the inverse shit center

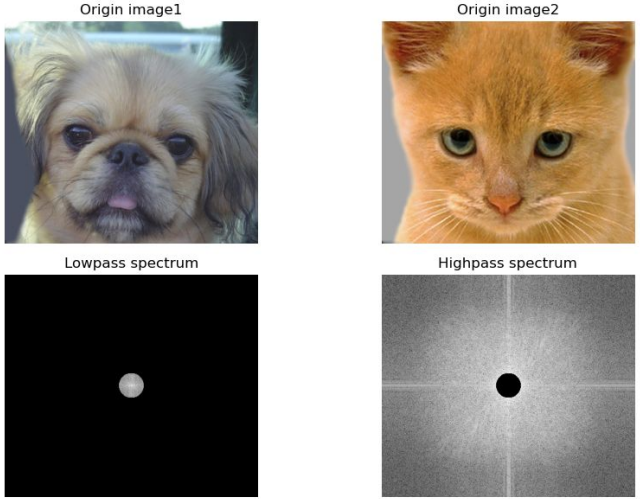
We use ifft2() function to compute inverse Fourier transform and ifftshift() function to inverse shift the center. Then we obtain the real part of result. Except generate the filtered version, we generate the spectrum

Step 4: Add low pass filtered version image and high pass filtered version image

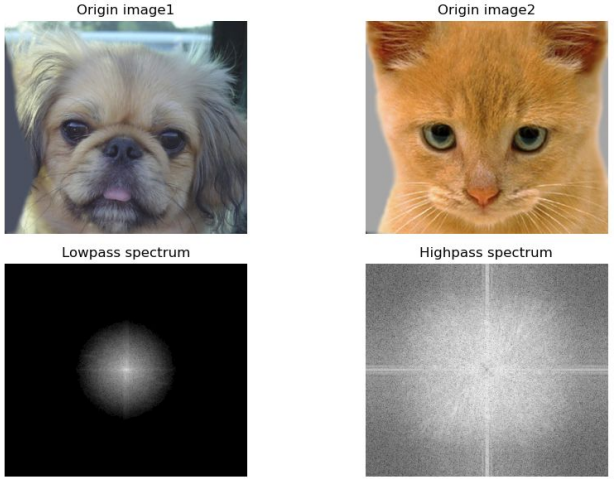
Experiment Result

Magnitude specturm of the pair 3: dog and cat (cutoff frequency is 20)

Ideal



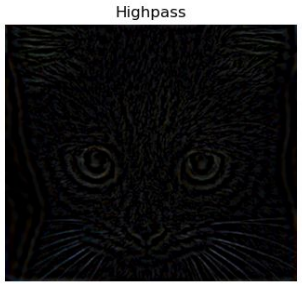
Gaussian



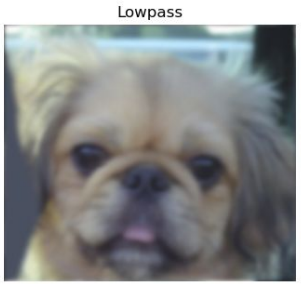
Experiment Result

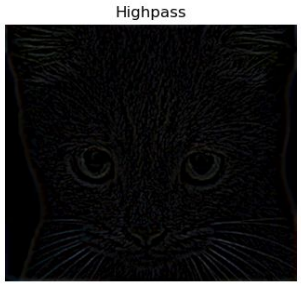
Hybrid result of the pair 3: dog and cat (cutoff frequency is 20)

Ideal



Gaussian

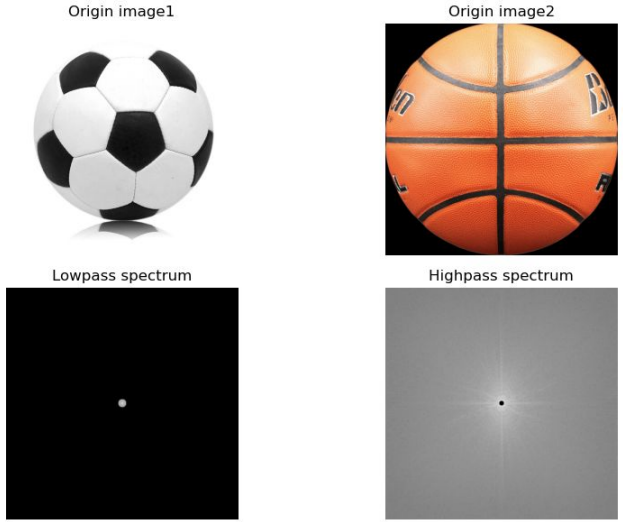




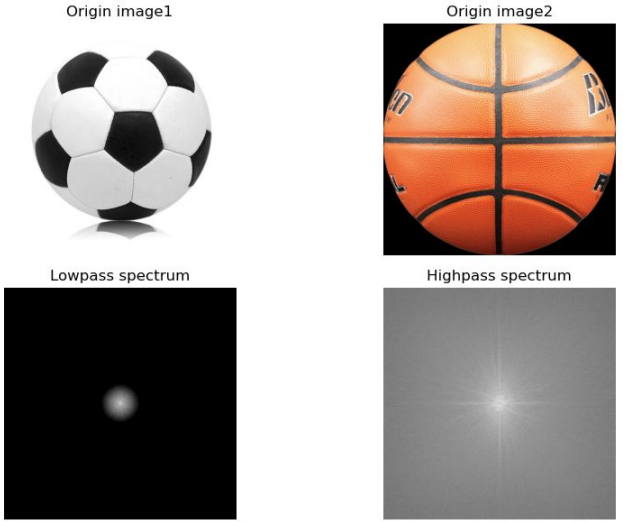
Experiment Result

Magnitude spectrum result of my images: basketball and football (cutoff frequency is 20)

Ideal



Gaussian

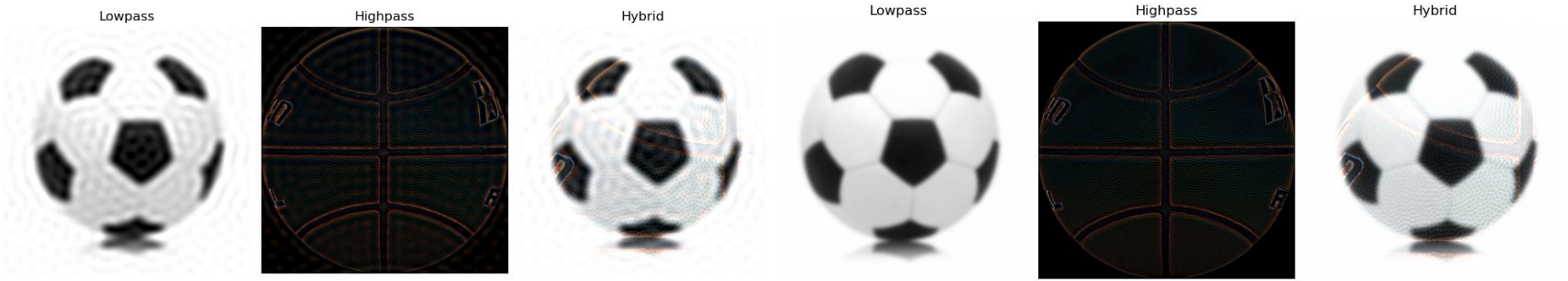


Experiment Result

Hybrid result of my images: basketball and football (cutoff frequency is 20)

Ideal

Gaussian



Discussion

We observe that the hybrid image result of Gaussian filter is more smooth than ideal filter. I think the reason is that the Gaussian low pass filtered version image is more smooth and the frequency does not cut directly when frequency is greater than cutoff frequency or less than cutoff frequency in Gaussian filter

Image pyramid

Corner is no longer a corner => Not invariant in scale

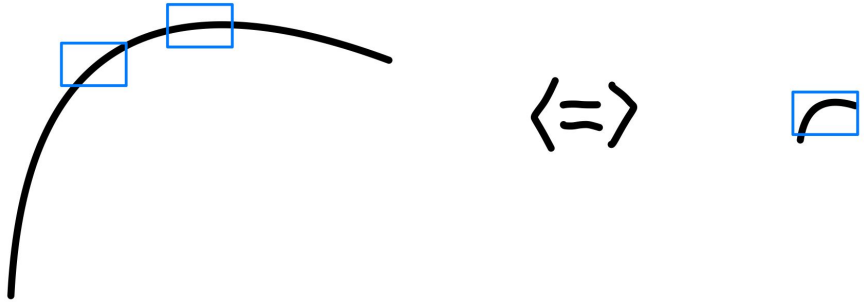


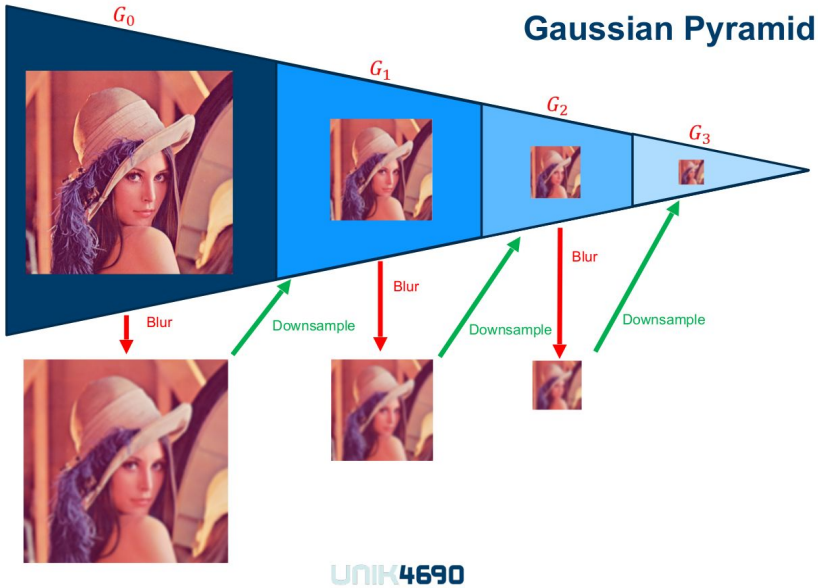
Image pyramid

repeat

filter

subsample

until min resolution reched



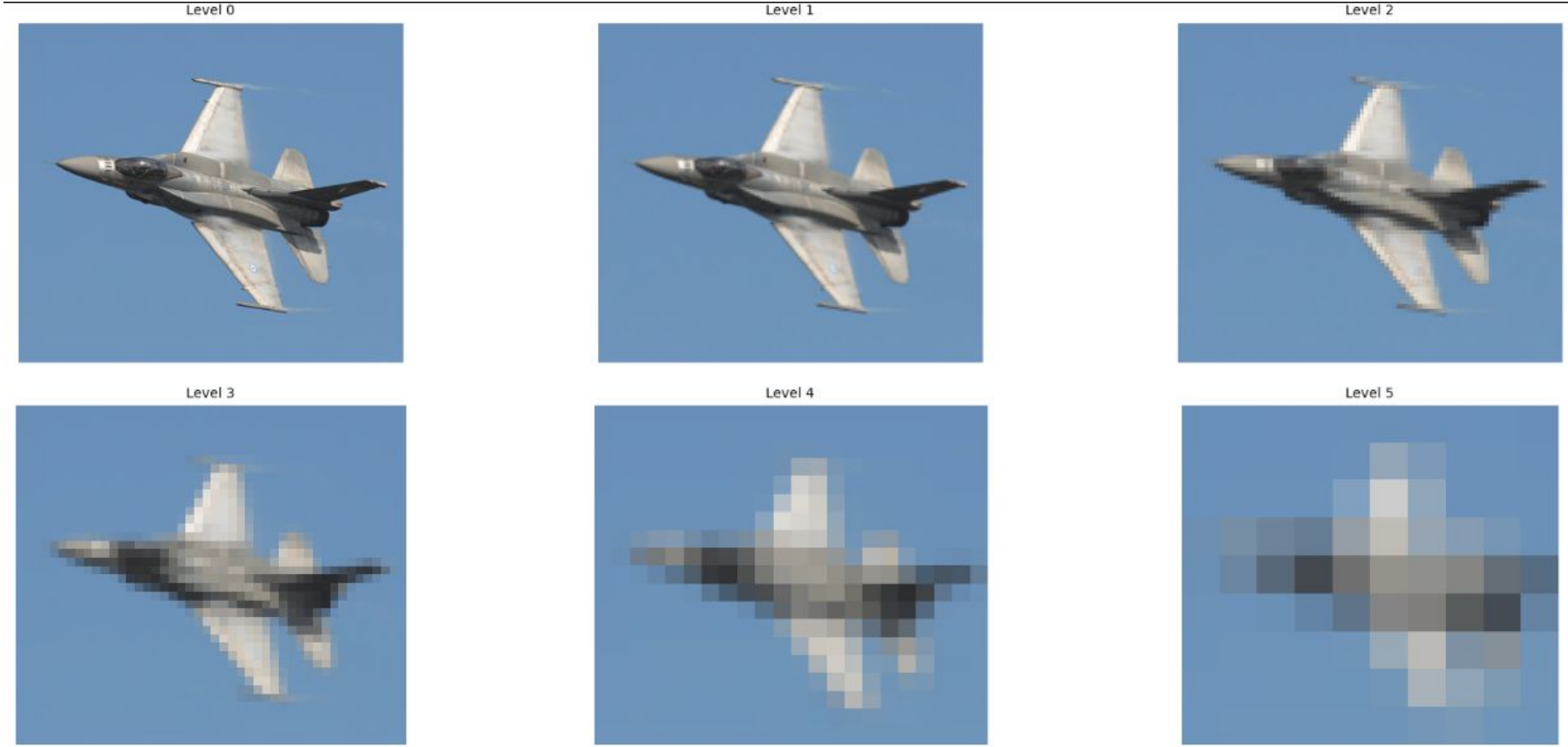

```
gaussian_blur_kernel = np.array([[1, 2, 1], [2, 4, 2], [1, 2, 1]])/16.0

for i in range(0, levels):
    # gaussion blur
    tmp = conv2d(tmp, kernel=gaussian_blur_kernel)

    # Subsampling, remove even
    tmp = tmp[::2, ::2]
    output.append(tmp)
```

Experiment result and discuession

Experiment Result



Level 0

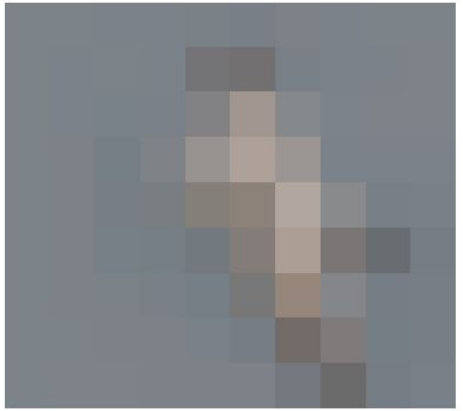
Level 1

Level 2

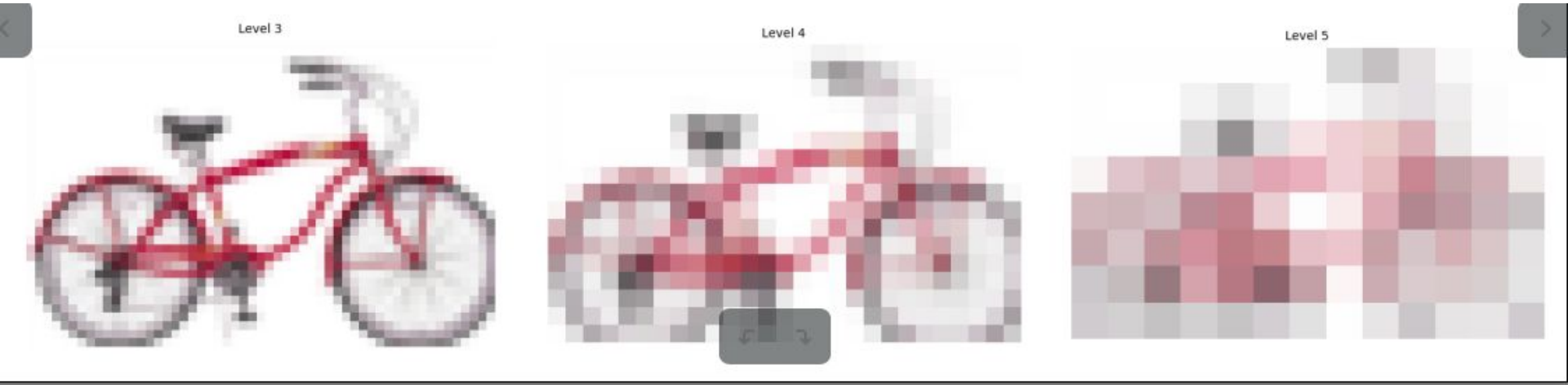
Level 3

Level 4

Level 5



scale down 越來越相似

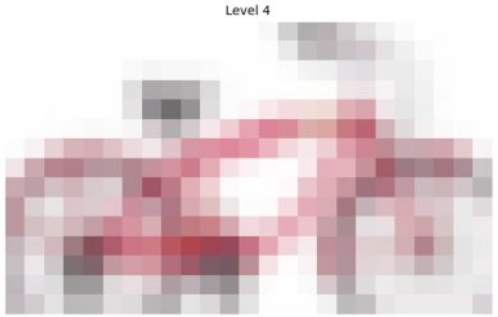


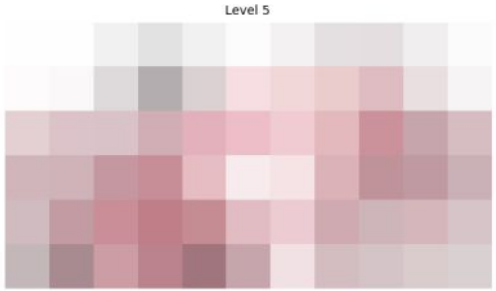
- High frequency components: pixel-to-pixel details
- Low frequency components: high-level structure
- What subsampling should do: remove pixel-to-pixel details, keep high-level structure.

Experiment Result

```
gaussian_blur_kernel = np.array([[1, 2, 1],
                                 [2, 4, 2],
                                 [1, 2, 1]])/16.0
# gaussian_blur_kernel = np.array([[1, 4, 7, 4, 1],
#                                  [4, 16, 26, 16, 4],
#                                  [7, 26, 41, 26, 7],
#                                  [4, 16, 26, 16, 4],
#                                  [1, 4, 7, 4, 1]])/273.0
```

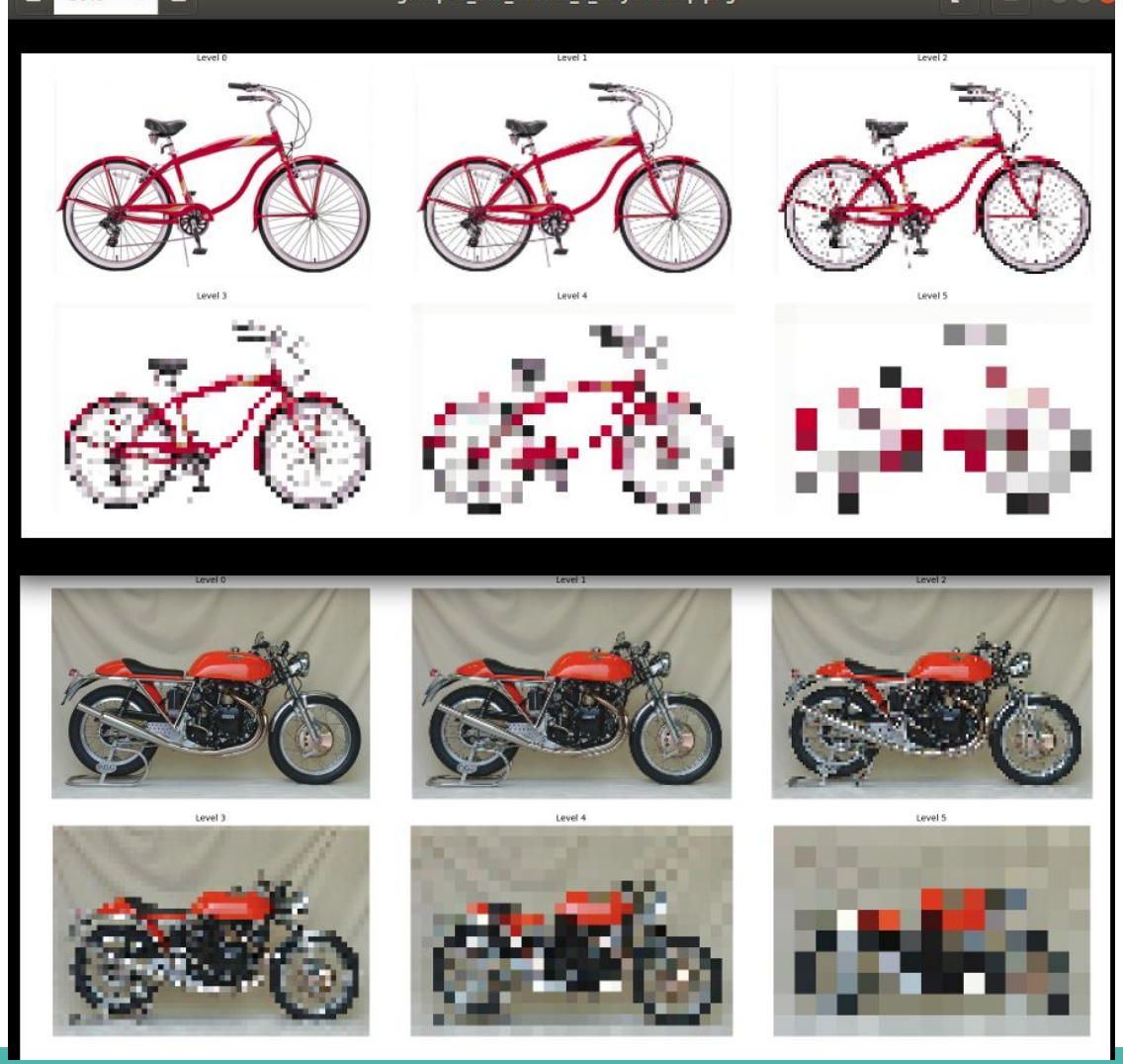
5x5





3x3

no gaussian blur

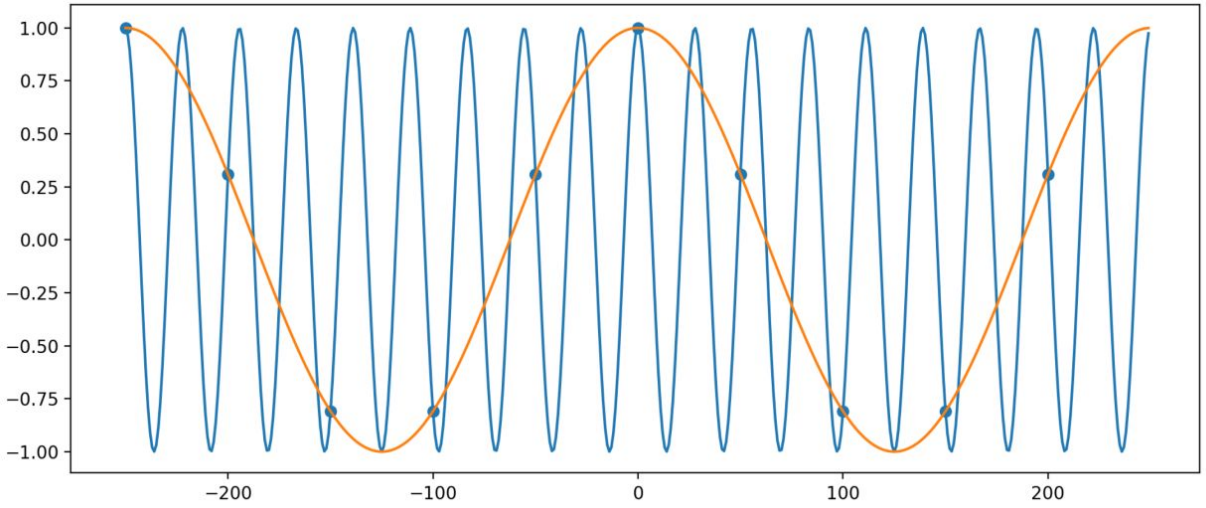


Aliasing

Nyquist sampling theorem: $2\nu_{max} < \nu_{sample}$

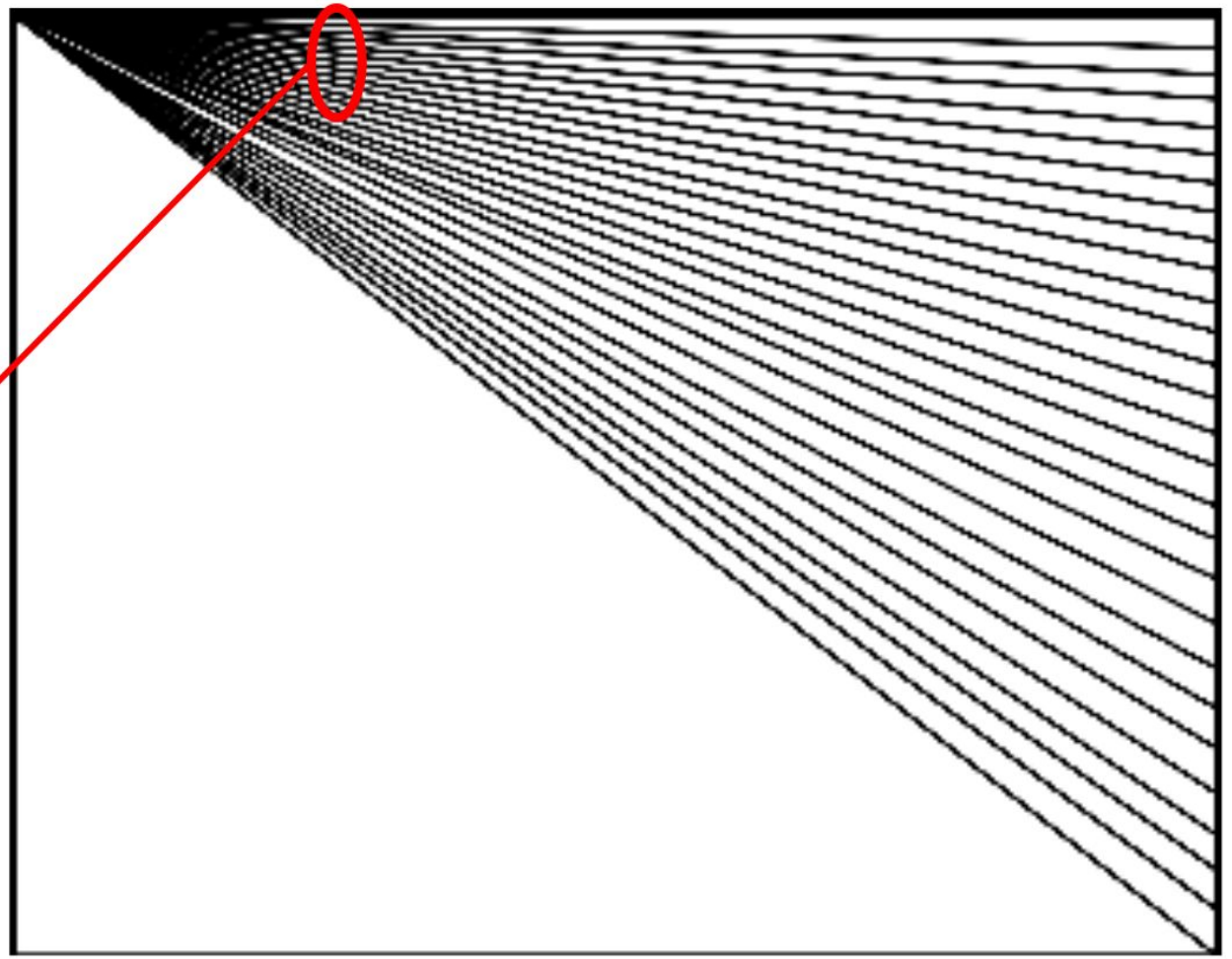
Specifically, in graphics:

- **Spatial aliasing**
- Temporal aliasing



Aliasing

Aliasing artifacts

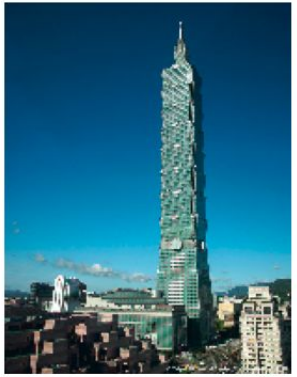


aliasing

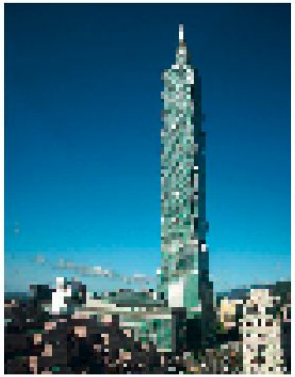
Level 0

Level 1

Level 2

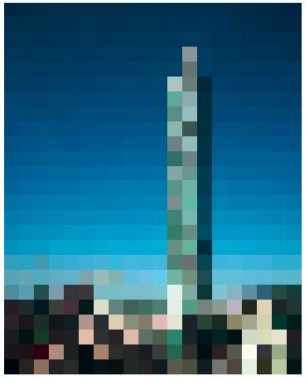


Level 3



Level 4

Level 5



gaussian blur + sub sampling

Level 0

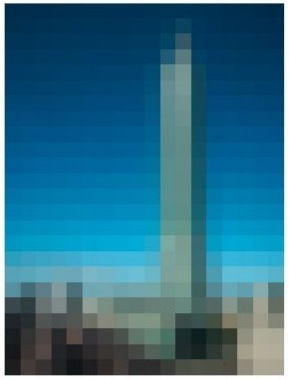
Level 1

Level 2

Level 3

Level 4

Level 5



Laplacian pyramid

$L_0 = g_0 - \text{EXPAND}(g_1)$

```
def apply_laplacian(reduced_list):
    output = []
    k = len(reduced_list)

    for i in range(0, k - 1):
        gauss = reduced_list[i]
        expanded_gauss = expand reduced_list[i + 1])
        while not expanded_gauss.shape[0] == gauss.shap
            gauss.shape[1]: …

        output.append(gauss - expanded_gauss)
```

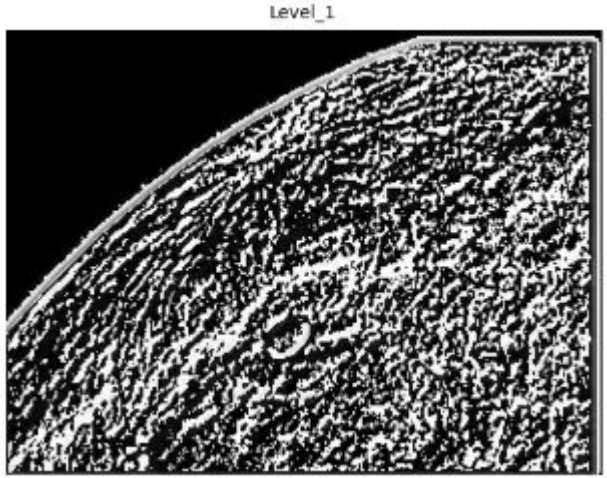
- Step 1: upsample and fill with 0s
- Step 2: Gaussian blur to interpolate
- Step 3: Scale correction
  - Gaussian blur is just weighted average
  - But we just introduced a bunch of zeros ==> need to scale up the resulting image

```
def expand(image):
    w_1d = np.array([0.25 - 0.4 / 2.0, 0.25, 0.4, 0.25, 0.25 - 0.4 / 2.0])
    W = np.outer(w_1d, w_1d)
    outimage = np.zeros(
        (image.shape[0] * 2, image.shape[1] * 2), dtype=np.float64)
    outimage[::2, ::2] = image[:, :]
    out = 4 * signal.convolve2d(outimage, W, 'same')
    return out
```

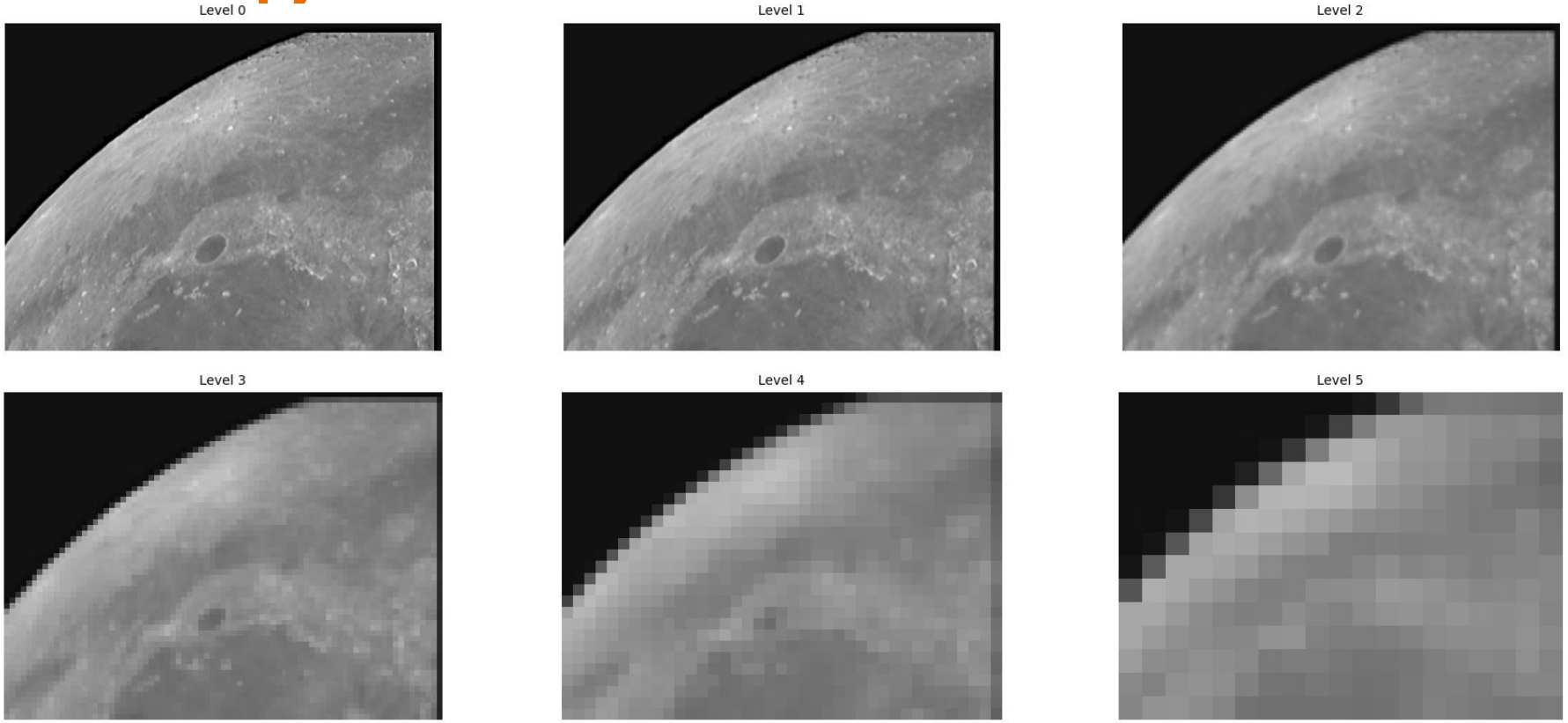
- expand(

)

=

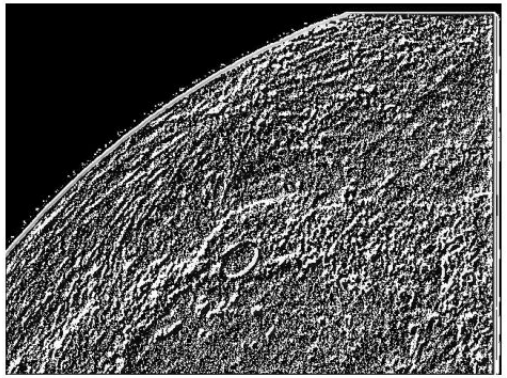


Gaussian pyramid



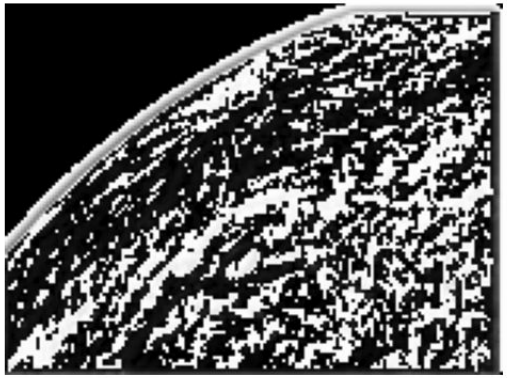
Laplacian pyramid - difference of gaussian

Level_0

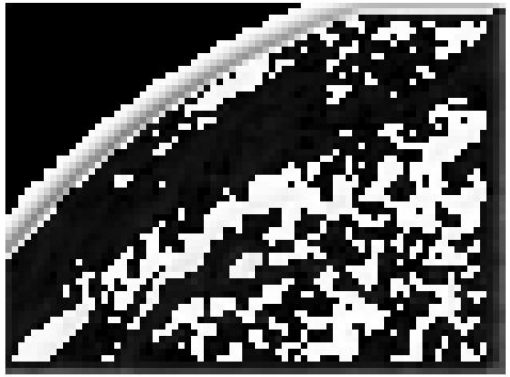


Level_1

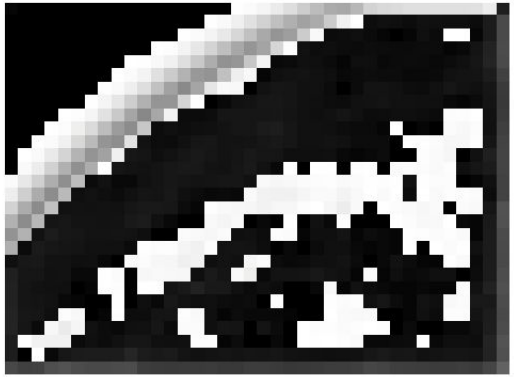
Level_2



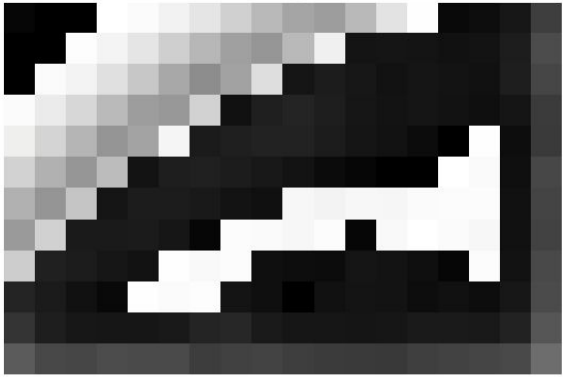
Level_3



Level_4



Level_5



Colorizing the Russian Empire

Introduction

Background to know

Sergei Mikhailovich Prokudin-Gorskii (1863-1944) was a photographer who, between the years 1909-1915, traveled the Russian empire and took thousands of photos of everything he saw. He used an early color technology that involved recording every scene's exposures onto a glass plate using a red, green, and blue filter. Back then, there was no way to print such photos, and they had to be displayed using a particular projector. Prokudin-Gorskii left Russia in 1918. The Library of Congress purchased his glass plate negatives that make it survived in 1948. Today, a digitized version of the Prokudin-Gorskii collection is available online.

Motivation

This assignment aims to learn to work with images by taking the digitized Prokudin-Gorskii glass plate images and automatically producing a color image with as few visual artifacts as possible. To do this, we will need to extract the three color channel images, place them on top of each other, and align them to form a single RGB color image.

Implementation Procedure

1. Use the Equation of Sum of Squared Differences (SSD) to represent alignment level. Find the lowest SSD to align two channels.

2. When processing big images with a vast database is computationally expensive, we would not want to waste our time.

3. Compare average SSD and edge SSD to detect the edge and then crop them.

4.Detect the error-cropped image, then fix it.

Implementation Procedure - SSD

Since we need to align three color channels, and their channels have similar brightness. So by **comparing pixel brightness of 2 channels**, use the Equation of Sum of Squared Differences (SSD) to represent alignment level. Find the **lowest** SSD to align two channels.

The Formula for Sum of Squares Is

For a set X of n items:

$$\text{Sum of squares} = \sum_{i=0}^{n} (X_i - \overline{X})^2$$

where:

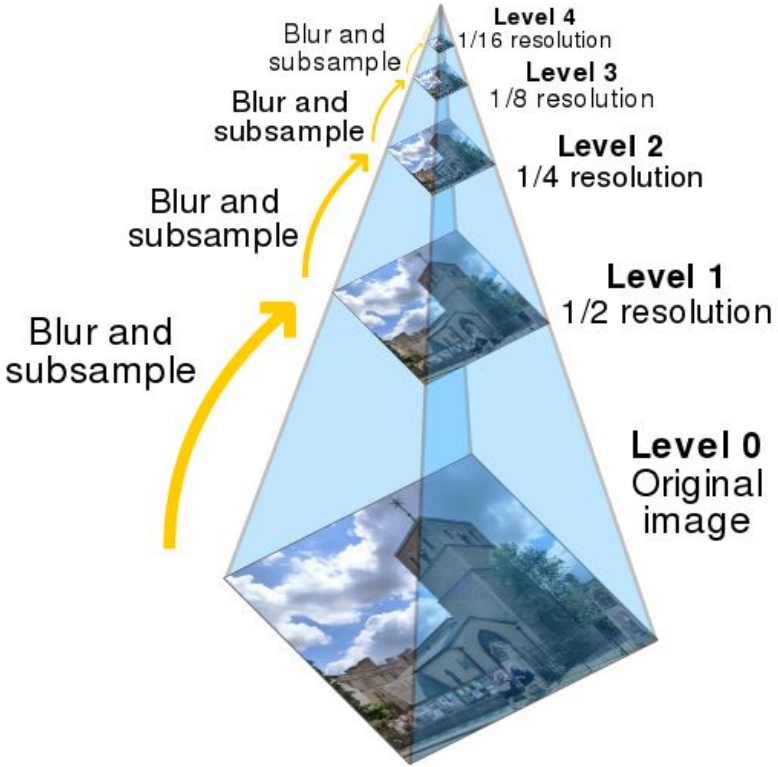
X_i = The i^{th} item in the set

$\overline{X}$ = The mean of all items in the set

$(X_i - \overline{X})$ = The deviation of each item from the mean

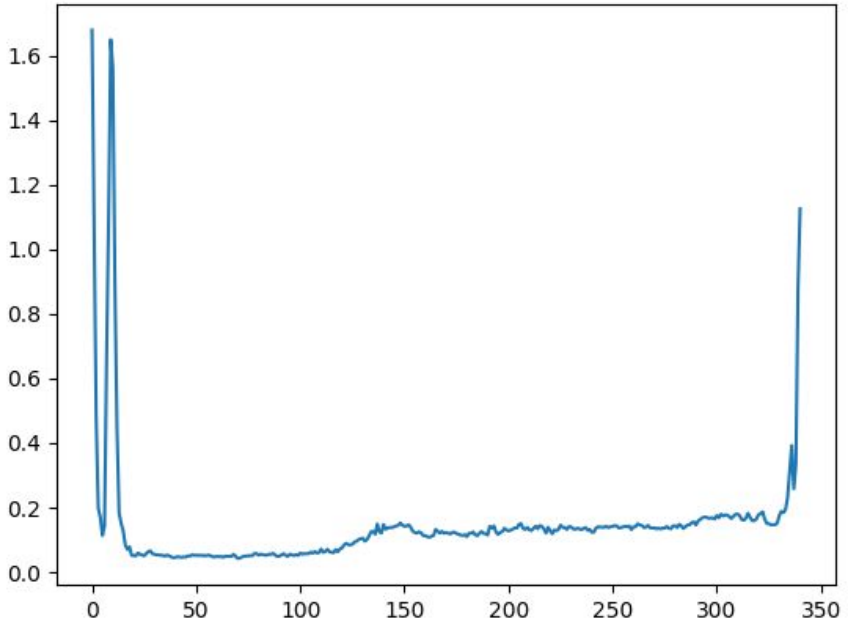
Implementation Procedure - use pyramid to reduce computational cost

When processing big images with a vast database is computationally expensive, we would not want to waste our time. Therefore, we apply the pyramid method to deal with the problem. We resize the photo and aligning from the small picture to the big picture. As for an image sized around 400x400, the shift is less than 15. Based on this, we determined our pyramid was aligning parameters and implementing our algorithm.



Implementation Procedure - crop the edge using SSD

We calculate the SSD of the 'cathedral.jpg' for each column and row as shown in the figures. We found that the edge SSD is usually much more extensive than the SSD in the middle of the picture. So we can compare average SSD and edge SSD to detect the edge and then crop them.



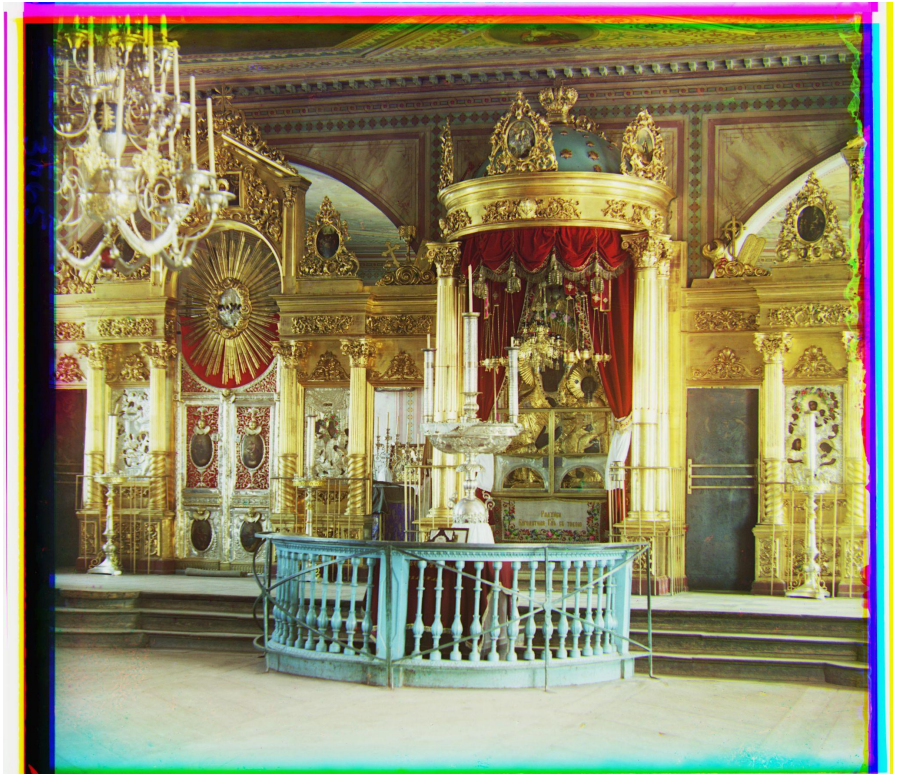
Implementation Procedure - detect the error-cropped image

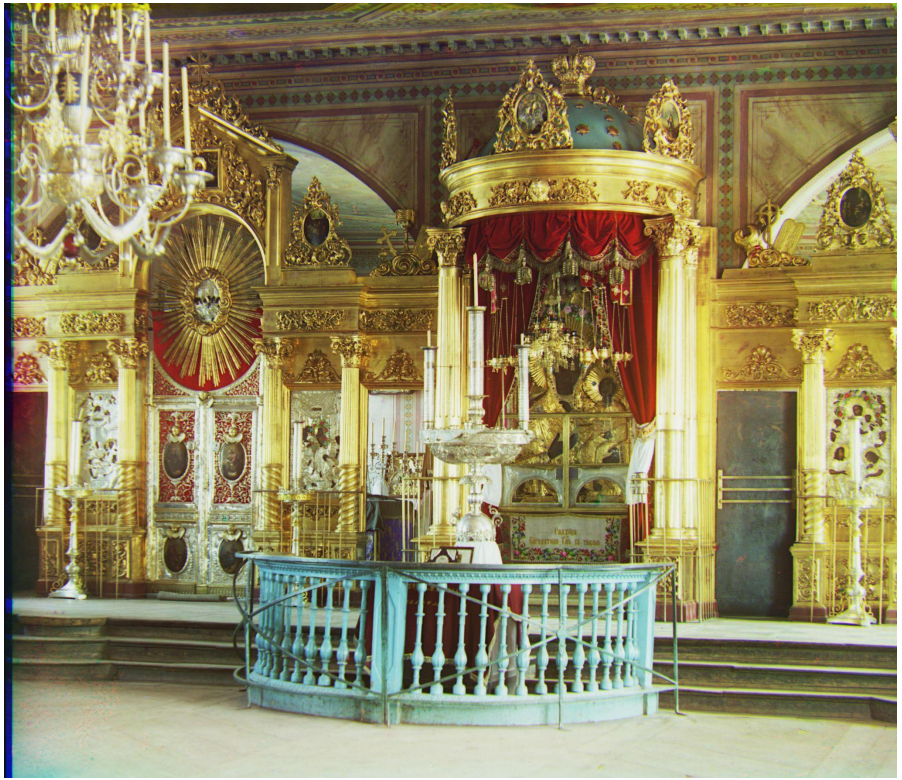
However, sometimes, using SSD to decide the edge of images is not much perfect. We can found that the image is overcropping than we hope. In this case, there is no better solution to use a hand-crafted method by directly cropping the 1% of the whole images, and the 1% comes from try and error.

Experiment-Before & Afrer Cropiing

Experiment-Before & Afrer Cropiing

Experiment-Before & Afrer Cropiing





Experiment-Before & Afrer Cropiing

Experiment-Before & Afrer Cropiing (detect the error-cropped image)

Experiment-Before & Afrer Cropiing

Experiment-Before & Afrer Cropiing

Experiment-Before & Afrer Cropiing

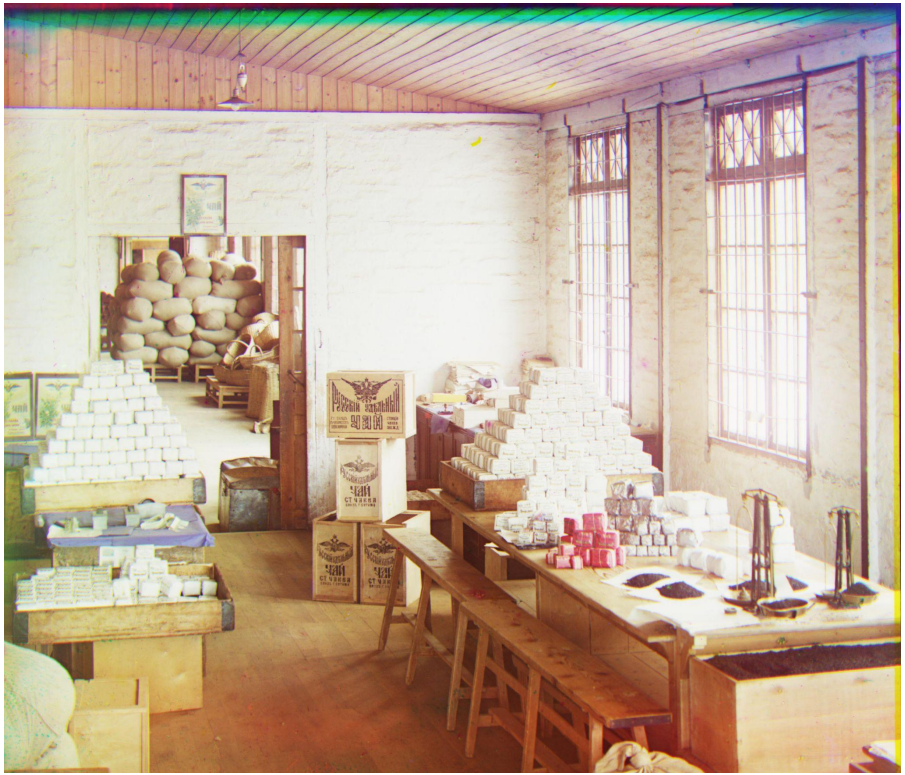
Experiment-Before & Afrer Cropiing

Experiment-Before & Afrer Cropiing

Experiment-Before & Afrer Cropiing

Experiment-Before & Afrer Cropiing

Experiment-Before & Afrer Cropiing



Discussion & Future work

- We use the **Equation of Sum of Squared Differences** (SSD) to represent alignment level. Find the **lowest** SSD to align two channels. However, processing big images with a vast database is computationally expensive; we would not want to waste our time.
- We found that the type of 'tif' input images took a long time to calculate the SSD for each RGB channel. For example, the 'melons.tif' image took us over 10 minutes to calculate the SSD because of its 3770*9724 high resolution. To deal with the problem, we a**pply the pyramid method.** We resize the photo and aligning from the small picture to the big picture. Based on this, we determined our pyramid was aligning parameters and implementing our algorithm.
- We found that the edge SSD is usually much more extensive than the SSD in the middle of the picture. So we can compare average SSD and edge SSD to detect the edge automatically and then crop them.
- However, sometimes, using SSD to decide the edge of images is not much perfect. We can found that the image is overcropping than we hope. In this case, there is no better solution to use a hand-crafted method by directly cropping the 1% of the whole images, and the 1% comes from try and error.

Work Assignment Plan

Hybrid Image
賴佑家

Image pyramid
李育靖

Colorizing the Russian Empire
陳平揚