CS180 Project 1

Xudong Chen, Student ID: 3040845876

September 9, 2024

1 Overview

In this project focusing on color channel alignment, I initially cropped the image to remove a 15% edge, which helps prevent black or white edges from interfering with the model's processing. Subsequently, I employed a combination of the Sobel Edge Detection algorithm and a Pyramid structure. This approach not only ensures precise color channel alignment but also enhances the speed of the algorithm when processing large images. Additionally, to address issues of color distortion, a color adjustment strategy was implemented to improve the quality of the aligned images.
In order to validate the generalizability and effectiveness of the algorithm, three additional images were selected for testing. This step is critical to ensuring that the color alignment technique performs consistently across diverse image types and real-world applications.

2 Sobel Edge Detection

The Sobel Edge Detection algorithm, introduced by Irwin Sobel in 1968, represents a pivotal advancement in image processing technologies. This method is now regarded as one of the most fundamental and widely-used techniques for edge detection in digital image processing. The Sobel operator distinguishes itself by its simplicity and efficiency, making it particularly effective for real-time applications.

2.1 How It Works

The Sobel operator employs two distinct 3×3 kernels, which are convolved with the original image to approximate the image's first derivatives in two orthogonal directions: horizontal and vertical. These kernels are designed to highlight gradients in these directions, which correspond to the edges within the image.
If we denote the source image by I, the results of the convolution of I with the Sobel kernels are two derivative images: G_x and G_y. Here, G_x contains the approximations of the horizontal gradients, and G_y contains the vertical gradients. These gradients at any point in the image are calculated as follows:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \cdot I, \; G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \cdot I \tag{1}$$

The magnitude of the gradient at each point gives the edge response at that point and can be calculated using following function

$$G = \sqrt{G_x^2 + G_y^2} \tag{2}$$

High magnitude of gradient represents the edge of items, after normalizing the result, I visualized one of the edge detection result as bellow

Figure 1: Sobel Scan Result

2.2 Compare Between Only NCC and Sobel Scan

Figure 2: Emir Without Sobel Scan

Figure 3: Emir With Sobel Scan

Figure 4: Harvesters Without Sobel Scan

Figure 5: Harvesters With Sobel Scan

2.3 Sobel Scan Speed Up

Without import any other libraries just 'Numpy', it would be too slow to use 'for' loops to scan a large image. To accelerate this process, I create a 6 layers matrix to help increase the speed - by moving around the calculated matrix and stack them together then use 'numpy.sum' can easily calculate the derivative result.
If the orange block is the pixel that need calculate the derivative, by moving around the matrix, stacking them and adding them together (For easier understanding, I didn't show out all of the moving matrix)

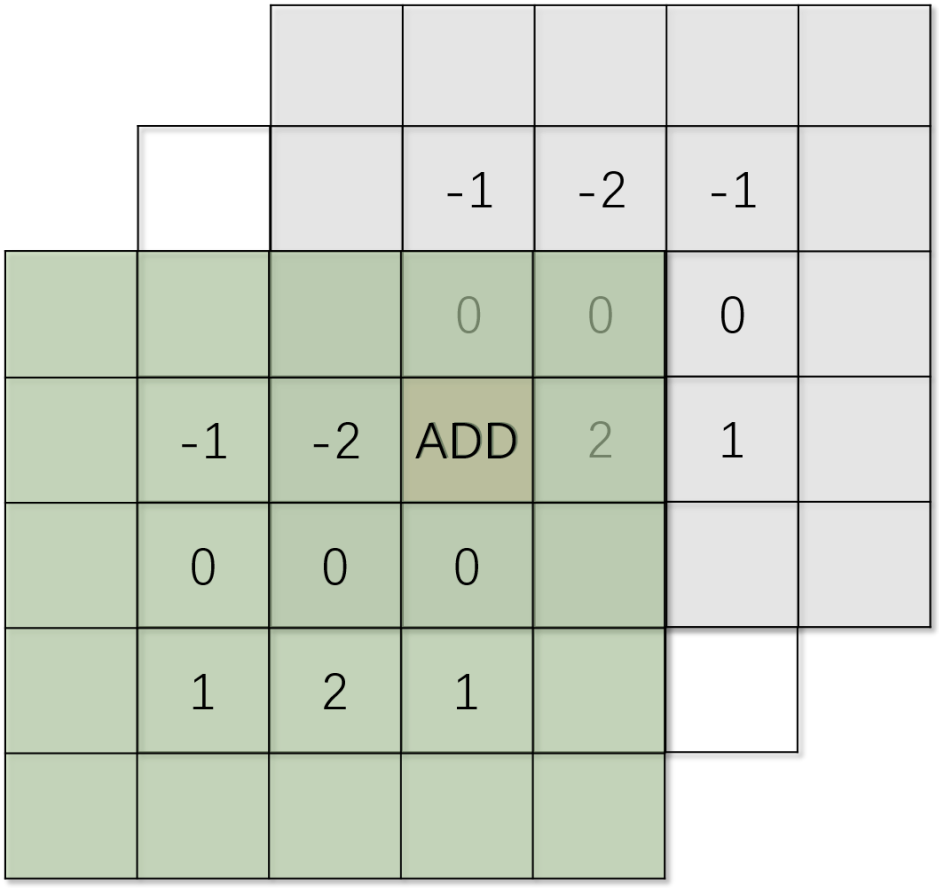

Figure 6: Sobel Speed Up

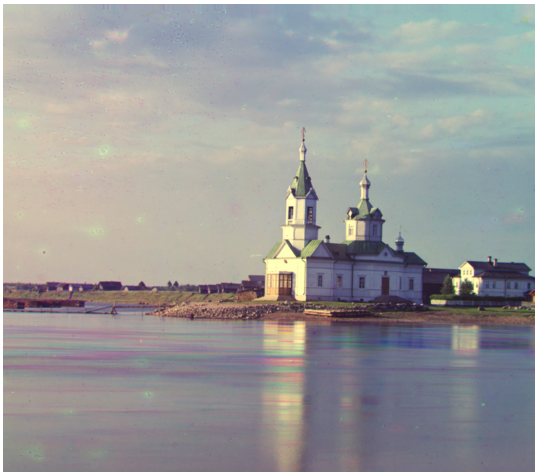

Figure 7: Church with Color Adjustment

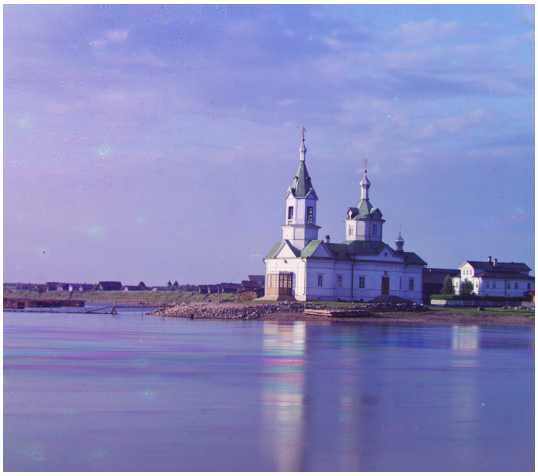

Figure 8: Church without Color Adjustment

Figure 9: Train with Color Adjustment

Figure 10: Train without Color Adjustment

3 Color Adjustment

I manually set the pixel value under a preset threshold as 0 and above as 255 and linearly adjust rest of pixels. After adjustment, I adjust the average pixel value in each channel and calculate the augment coefficients and apply it to each channel. The coefficients' function is listed as follow:

$$\text{coef} = \frac{\text{average}(I)}{\text{average}(C)} \tag{3}$$

The adjustment method is listed as follow:

$$\text{adjust C} = C \times \text{coef} \tag{4}$$

The image is placed above the text

4 Results

4.1 Low Quality Images

Figure 11: cathedral.jpg

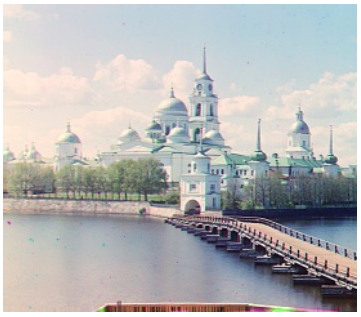

Figure 12: monastery.jpg

Figure 13: tobolsk.jpg

4.2 High Quality Images

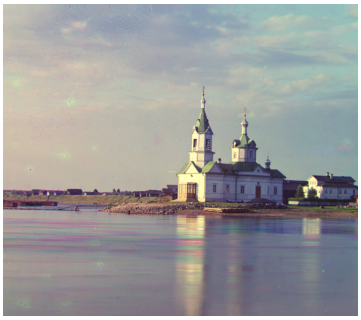

Figure 14: church.jpg

Figure 15: emir.jpg

Figure 16: harvesters.jpg

Figure 17: icon.jpg

Figure 18: lady.jpg

Figure 19: melons.jpg

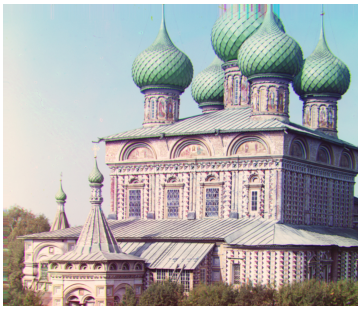

Figure 20: onion church.jpg

Figure 21: sculpture.jpg

Figure 22: self portrait.jpg

Figure 23: three generations.jpg

Figure 24: train.jpg

4.3 Self-Select-Images