CS270 Digital Image Processing

Spring 2021 & Homework 1

Chuanyang Hu

2020233252

March 31, 2021

Problem 1

A. Description

In this question, we first request that you can improve the quality of one given image using image enhancement methods implemented by yourself. For example, you could improve the image contrast by adjusting the histogram or try some image filters. As shown in Fig.1, (a) shows the original image which looks foggy; (b) gives one example result which seems clearer after improving the contrast.

Then, we need you to impair the image quality of one normal image (see Fig.2), which is contrary to the task above. You would make the clear image foggy after changing the contrast. By accomplishing these 2 tasks, you will understand better the process of image enhancement.

B. Solution

1. Defogging

We solve the defogging problem by using Kaiming method: Single Image Haze Removal Using Dark Channel Prior. According to the prior of computing graphicx, a foggy image can be represented by the Equation.1.

$$I(x) = J(x)t(x) + A(1 - t(x)) \tag{1}$$

$I(x)$ represented the observed image (foggy image), $J(x)$ represented the clear image (ground-

truth), A represented the atmosphere light and $t(x)$ represented the transmission ratio. Therefore, we can restore the clear image by estimating A and $t(x)$.

According to the dark channel prior, In most of the non-sky patches, at least one color channel has very low intensity at some pixels. In other words, the minimum intensity in such a patch should has a very low value. As a result, we can get the Equation.2

$$J^{dark}(x) = \min_{c\in\{r,g,b\}}(\min_{y\in\omega(x)} J^c(y)) \to 0 \tag{2}$$

According to Equation.1, we can get the restoration Equation of $J(x)$:

$$J(x) = \frac{I(x) - A}{max(t(x), t_0)} + A \tag{3}$$

The t_0 in Equation.3 guarantees the correctness (the equation is invalid if $t(x) \to 0$) and we set t_0 to 0.1.

$$\begin{aligned} \frac{I(x)}{A} &= \frac{J(x)t(x)}{A} + (1 - t(x)) \\ \min_c(\min_{y\in\omega(x)}(\frac{I^c(y)}{A^c})) &= \min_c(\min_{y\in\omega(x)}(\frac{J^c(y)}{A^c}))t(x) + (1 - t(x)) \end{aligned} \tag{4}$$

The Equation.4 is to estimate the transmission ratio. Meanwhile, according to Equation.2, we can get Equation.5.

$$t(x) = 1 - \min_c(\min_{y\in\omega(x)}(\frac{I^c(y)}{A^c})) \tag{5}$$

$$\widetilde{t}(x) = 1 - w\min_c(\min_{y\in\omega(x)}(\frac{I^c(y)}{A^c})) \tag{6}$$

In practice, even in very sunny weather, the transmittance of the atmosphere will not be completely 1, and the blur will still exist when we look at distant objects. (Excessive deblurring will cause the image to be unnatural), so in order to restore a close-to-natural image, we need to retain a certain degree of blur, so we can add a coefficient w before the blur rate of Equation A and we set w to 0.95. The Equation.6 is the estimation of $t(x)$.

Finally, according to the method proposed by Kaiming, first select the top 0.1% of the pixels in the dark channel image, and the dark channel prior conditions ensure that the selected pixels will

not be interfered by the natural image scene, and then find the corresponding 0.1% in the original image of the brightest point of each channel of pixels, which is used as the estimated value of atmospheric light intensity A.

The lines 34 to 61 of code are calculating the dark channel image. The lines 63 to 74 of code are estimating the atmosphere light. The lines 76 to 78 of code are estimating the transmission ratio. The lines 80 to 85 of code are recovering the clear image. The lines 87 to 93 are the main ordered process of defogging. I have explained all the details of this algorithm above and the code is just a implementation of the algorithm.

2. Fogging

According to the Equation.1, we only need to generate the value of A and $t(x)$ which can help us to add fog in image. As a result, we randomly generate the value by numpy.random.uniform to get A and $t(x)$ and calculate the fogging image $I(x)$.

The lines 95 to 100 of code are adding fog using the algorithm mentioned above which just use few operations of numpy.

3. Summary

The defogging and fogging result are Figure.1

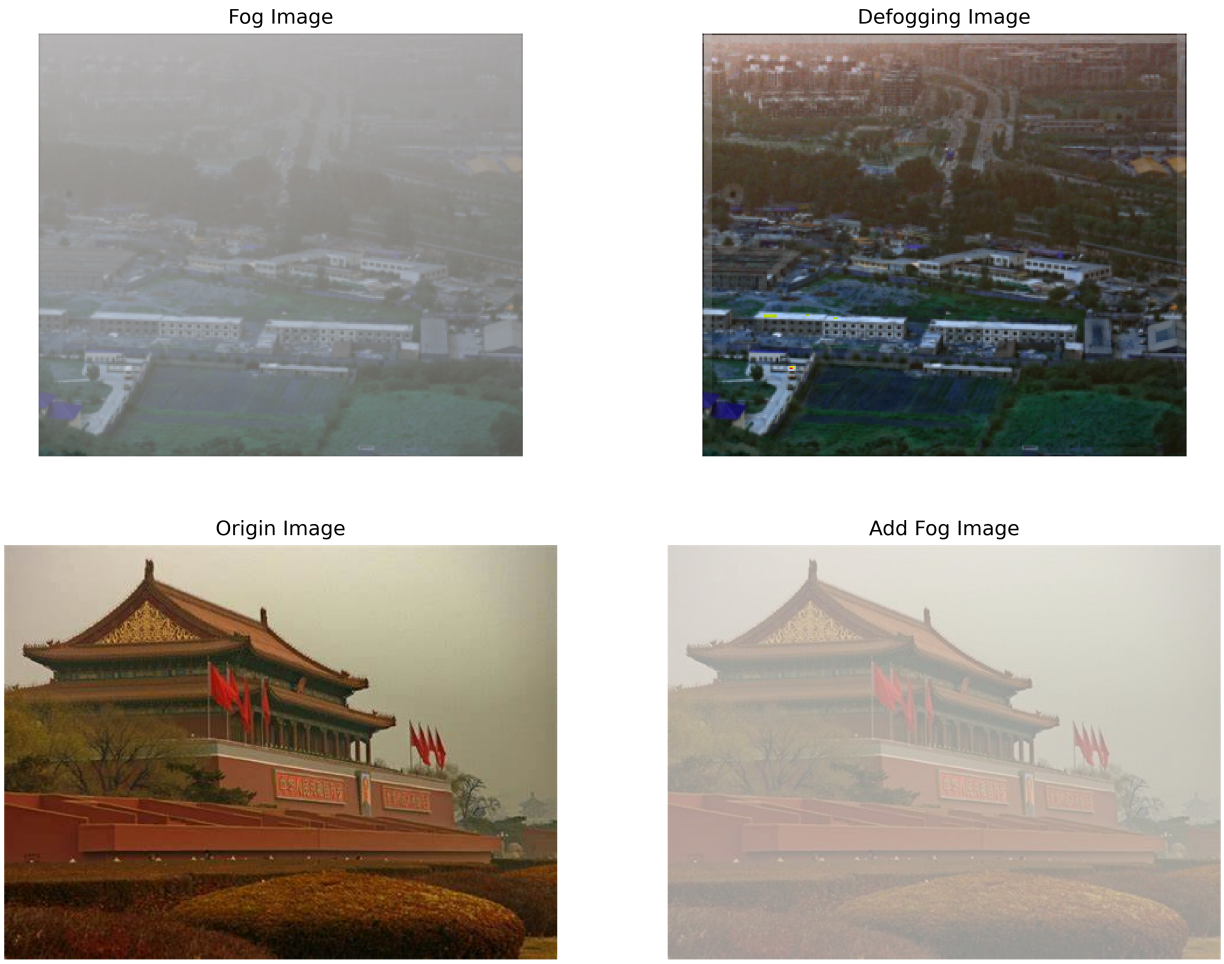

Figure 1: Processing Results

Problem 2

A. Description

Image stitching is the process of combining multiple photographic images with overlapping fields of view to produce panoramic or high-quality views.

B. Solution

We use SIFT to extract features from images.

- Extremum detection in scale space: Search for images in all scale spaces, and identify potential points of interest that are invariant to scale and rotation through Gaussian differential functions.

- Feature point location: At each candidate location, a fine fitting model is used to determine the location scale, and the selection of key points is based on their degree of stability.
- Feature direction assignment: Based on the local gradient direction of the image, one or more directions are assigned to each key point position. All subsequent operations are to transform the direction, scale and position of the key point to provide the invariance of these features.
- Feature point description: In the neighborhood around each feature point, measure the local gradient of the image on a selected scale, and these gradients are transformed into a representation that allows relatively large local shape deformation and illumination transformation.

Then we use the brute force search method to match the features, and at the same time, we need to calculate the two points closest to the feature of each point in the original image in the target image which means we will compare each combination of points.

Let us define point A in original image and points B_1, B_2 in target image. Meanwhile, we assume that the features extracted from point B1 and point B2 using SIFT are the closest and the second closest to the feature of point A. After running SIFT algorithm and feature matching, the feature distance between A and B_1 is D_1 and the feature distance between A and B_2 is D_2. Therefore, according to Equation.7, we can get the discriminant value D_v.

$$D_v = \frac{D_1}{D_2} \tag{7}$$

If D_v less than the threshold T, we will adopt the match between point A and point $B1$. Otherwise, we will think this match is not good and regard it as mismatch because the closest match is very different from the second closest match which means it maybe noisy.

Then we use the following steps to do the transformation:

- Perspective transformation
- Calculate homography matrix
- Wrap Perspective
- Stitch images

The essence of the homography is the simple 3×3 matrix called the homography matrix.

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

if we take a point $A(x_1, y_1)$ in the first image we can use a homography matrix to map this point A to the corresponding point $B(x_1, y_1)$ in the second image.

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{8}$$

To find this transformation matrix, we need to extract coordinates of a minimum of 4 points in the first image and corresponding 4 points in the second image. These points are related by homography so we can apply a transformation to change the perspective of the second image using the first image as a reference frame. Finally, we can stitching the two images. The Figure.2-Figure.9 are the matching results.

Figure 2: Matching points

Figure 3: Stitching Image

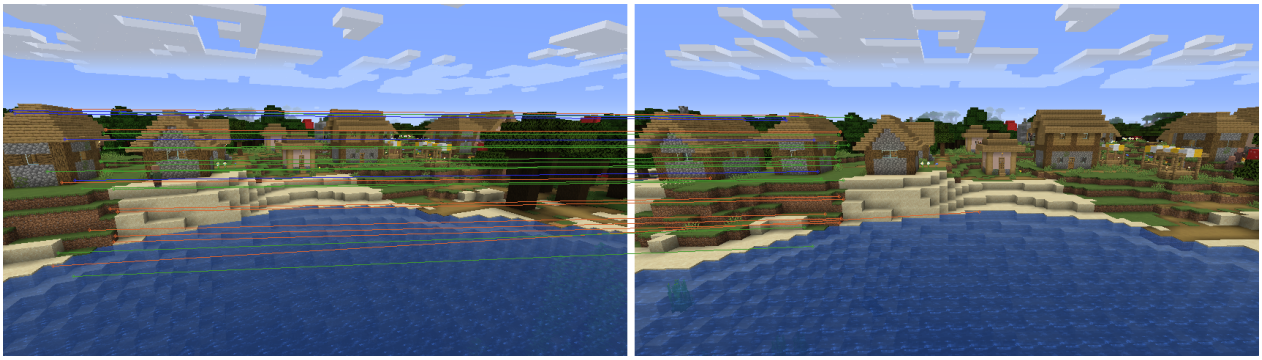

Figure 4: Matching points

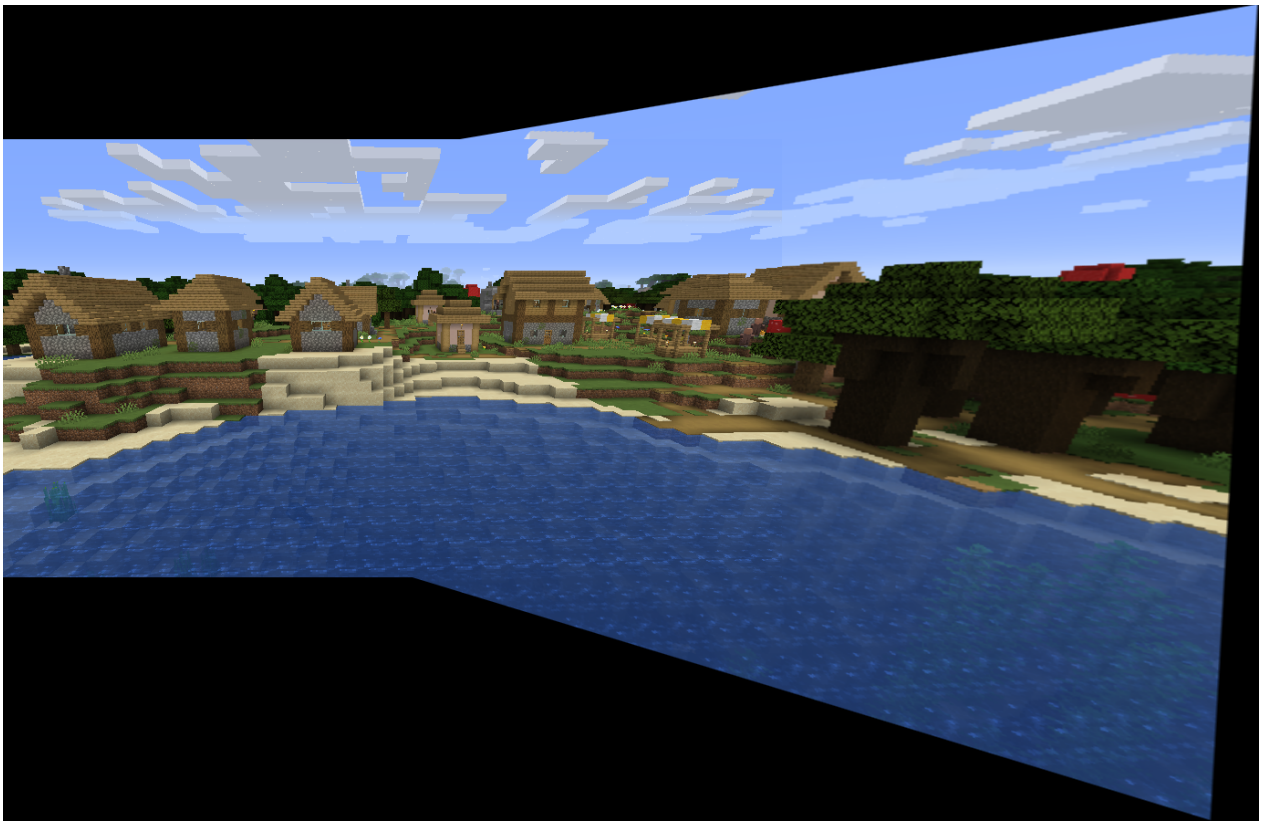

Figure 5: Stitching Image

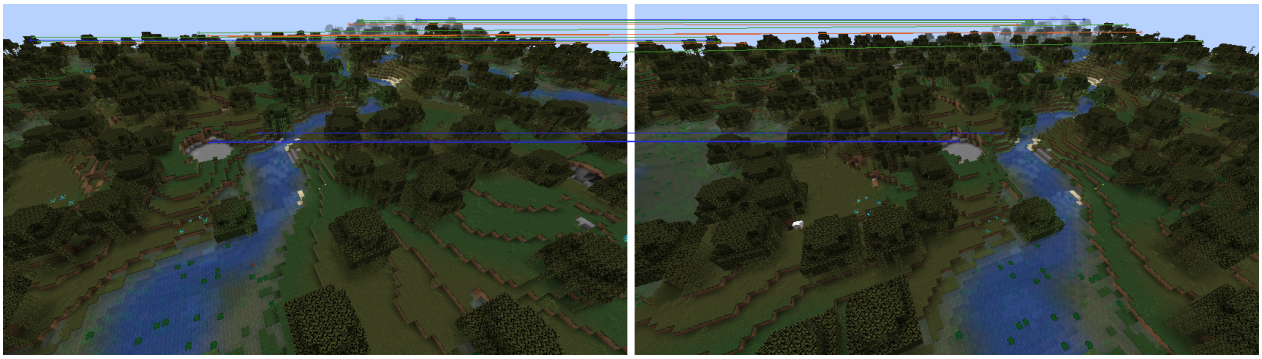

Figure 6: Matching points

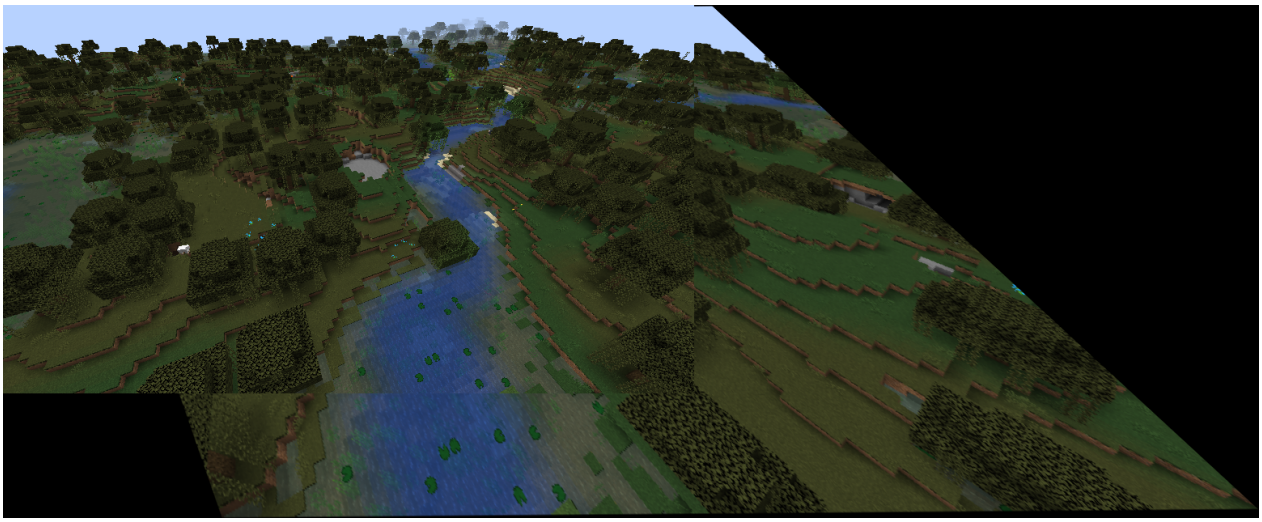

Figure 7: Stitching Image

Figure 8: Matching points

Figure 9: Stitching Image

Problem 3

A. Description

In this question, please implement color transfer between images using MATLAB or other languages. Here is a simple algorithm based on statistical analysis to impose one image's color characteristics on another. It chooses an orthogonal color space without correlations between axes to simplify the color modification process. Ruderman et al.'s perception-based color space $l\alpha\beta$ minimizes correlation between channels for many natural scenes. This space is based on data-driven human perception research which assumes that the human visual system is ideally suited for processing natural scenes. The mean and standard deviations along each of the three axes suffice to make an image take on another image's color characteristics. The color transfer result's quality depends on the similarity of the images in composition. Sometimes it fails and you can try to manually select two or more pairs of clusters in $l\alpha\beta$ space for transformation. Another possible extension would be to compute higher moments such as skew and kurtosis, which are respective measures of the lopsidedness of a distribution and of the thickness of a distribution's tails. Imposing such higher moments on a second image would shape its distribution of pixel values along each axis to resemble the corresponding distribution more closely in the first image.

B. Solution

We can just follow the algorithm in the instruction PDF. First, we use cv2 to read the original image and convert to RGB space (cv2 is BGR format). Then we transform the RGB image to LMS space by matrix multiplication. Finally, we transform LMS image to LAB space also by matrix multiplication. But before we do the final step, we should add epsilon to the image to make sure that we will not get -Inf during logarithmic operation. After adding epsilon, a log operation is applied to convert the data to logarithmic space. The images in $l\alpha\beta$ color space are Figure.10

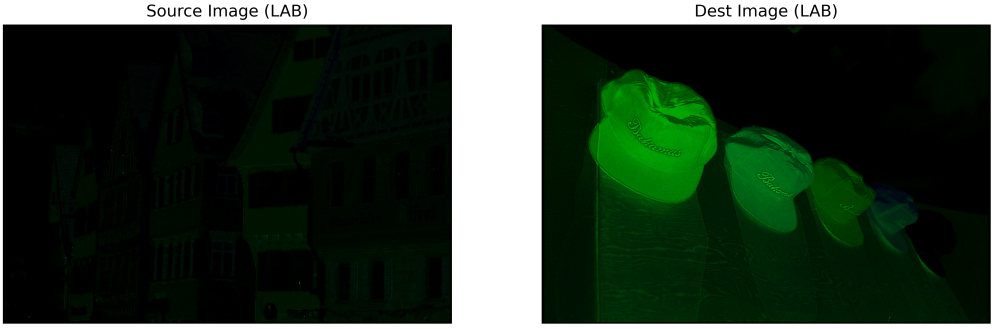

Figure 10: Images in $l\alpha\beta$ color space

Next, we visualize each channel of source image in $l\alpha\beta$ color space (Figure.11)

Figure 11: Channel visualization

Then, we implement the statics and color correction in $l\alpha\beta$ color space. The lines 59 to 72 of code is the implementation following the Equation.9 and the other code is just to do some matrix multiplication and visualization (matplotlib). The lines 20 to 51 of code is all the coefficient of transformation matrix.

$$I_{correct} = (I_{src} - mean_{src}) \cdot \frac{std_{dst}}{std_{src}} + mean_{dst} \tag{9}$$

The result of color correction is Figure.12

Figure 12: Color correction

Therefore, we can do the color transfer and apply to other images. Figure.13-15 are the transfer results of required images. Figure.16 is the result of the images selected by me. You can see that we have transferred the light tones of the target image to the first darker image.

Figure 13: Color transfer

Figure 14: Color transfer

Figure 15: Color transfer

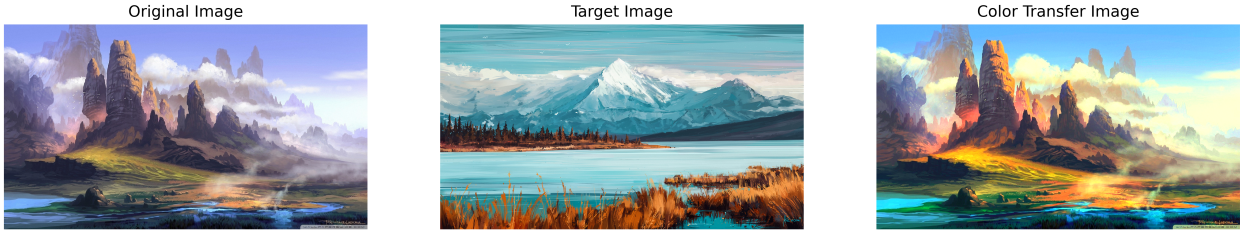

Figure 16: Color transfer