1 Introduction

The purpose of the report is to simply illustrate the method I used for this assignment. For details, please reference the code.

2 Method

Naive Method

The naive method is simply crop the image into 3 parts vertically and stack them for a color image. This method is easy to implement but can cause a huge misalignment.

NCC

I choose Normalized cross-correlation (NCC) to solve the misalignment issue. The NCC is calculated as:

1. For two matrix, extract mean respectively to get new matrix, matrix A, matrix B.
2. Divided by their $\mathbb{L}2$ norm respectively. $A = \frac{A}{||A||_2^2}, B = \frac{B}{||B||_2^2}$
3. Calculate NCC as their dot product:

$$NCC = A \cdot B$$

I used greedy search to find best index to chop the image for two image A,B. For any pixel, I search 20 pixels from each direction and the best chop index should give me the smallest

NCC score. The detail of the NCC algorithms has shown below:
Result: Return the best chop index i,j
initialization Construct **index stack** for candidate chop index along 4 directions
Initialize **Score** as -1;
for *i,j in index stack* **do**
    Chop image B based on i,j. Calculate the NCC-A-B score between image A and image B. ;
    if *NCC-A-B > score:* **then**
        score=NCC-A-B;
        Best-Chop-index=[i,j];
    end
end

Algorithm 1: NCC Alignment

After align the Red image and Green image to Blue image respectively, I stacked three image together to the final image.

2.1 Crop Border

Without cropping the border, the border of the image has unpleasant result. In order to fix this issue, I try 3 way to crop:

1. Crop the image before split.
2. Crop the image after alignment.
3. Crop vertical upfront and chop horizontal after alignment or the other way around.

After 3 different try, I found chop vertical upfront and chop horizontal after alignment give me best result, which fixed border issue and reserve as much as information of original image as possible. For parameter to chop, I used grid search based on different set of parameters.

2.2 High resolution image

For high resolution image, I tried two different method: The first method is, I resized image using scipy.misc.imresize to form 4 different images, which are the 100%, 50%, 25%, and 12.5% of original image in terms of size. Start from the smallest image, I search the best chop point i,j from rectangle window size of 20*20. Times i, j by scale index to get new start index of upper level image. So we can calculate the best index recursively. While this method took too long when the image matrix is too big to calculate matrix norm. Align a single tif file could take up to 4 minutes.
Another method is scaled down to 10 percent of original image and find the best chop index i,j. Then times i, j by 10 to get chop index of high resolution image. This method is much more simple than previous one and much faster, which took less than 10 second. Also the quality is surprisingly high! The result has shown in the report.

Potential Improvement

Due to time issue, I just tried method I mentioned above. But there are some potential improvements. For example, when we scaled down the image, the brightness and contrast may changed due to size change. We can adjust brightness and contrast respectively. For border crop, I just manually choose the parameter based on observation. A more sophisticated method maybe can detect the border automatically.

3 Additional information

The **align.py** contains the code for alignment of small image. **alighhigh1.py** contains the code for alignment of large image using recursion. **alighhigh2.py** contains the code for alignment of large image using one time solution. which is I finally choice to output the high resolution image.