GAZI UNIVERSITY

FACULTY OF ENGINEERING AND ARCHITECTURE

COMPUTER ENGINEERING

CENG471 INTRODUCTION TO IMAGE PROCESSING ASSIGNMENT I

Prokudin-Gorskii Images

Prepared by

Selçuk SOLMAZ

161180052

Instructor

Dr. DUYGU SARIKAYA

ANKARA, NOVEMBER 2020

1. Prokudin-Gorskii

Firstly, I searched Prokudin-Gorskii. I would research how I should use these image. Prokudin-Gorskii aimed to produce a colored image by passing a image taken in black and white through three filters (blue, green, red) [1].

- Firstly, I converted the given image to BGR2GRAY format with the cvtColor function.
- Then, I split the image in BGR2GRAY format into 3 parts. Used len(image/3) function.
- For the image is BGR from top to bottom, I divided the pictures as blue, green and red.
- I coded splitting the image into three parts with the Image_Crop (image) function.

Figure 1. Cropping Blue

Figure 2. Cropping Green

Figure 3. Cropping Red

- After splitting the image, we can apply the NCC algorithm.
- I've set the range to [-20,21].
- I find the best NCC with the function NCC_Algorithm (color1, color2, distance).
- color1 image to roll and color1 is moved over color2 image.
- distance is 20. So, range is [-20,21].
- The main purpose of the NCC algorithm is to reach the highest.
- color1 rolled to color2.
- Then;

  NCC = np.sum((moved_color1/np.linalg.norm(moved_color1)) * (color2/np.linalg.norm(color2))) used. The total number of NCC's is found. I used norm function because I didn't want high values [2].

  For example;

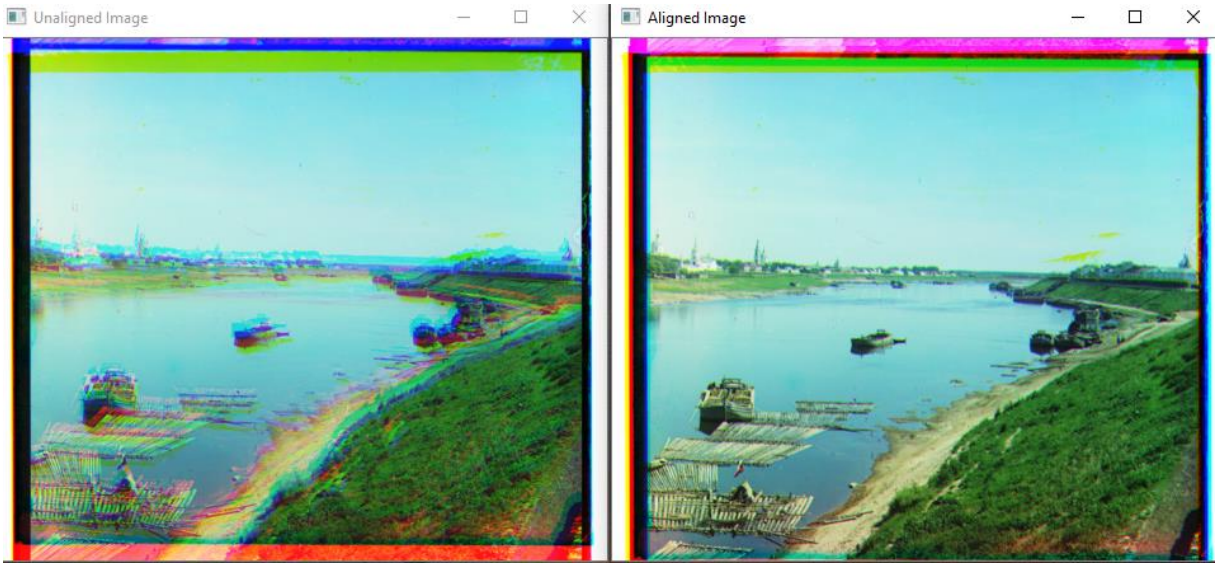
```
                        0.29409101429988993
0.134956389438551       0.295342644352904
0.13578549033264375     0.2965743789271061
0.13661929030085865     0.29728923043046784
0.13762596797430537     0.29913441792280004
0.13891856414119177     0.30191256142965744
0.13999813375889253     0.3021041155947017
0.1407911749488492      Align : [12, 4]
```

Figure 4. Total NCC Example

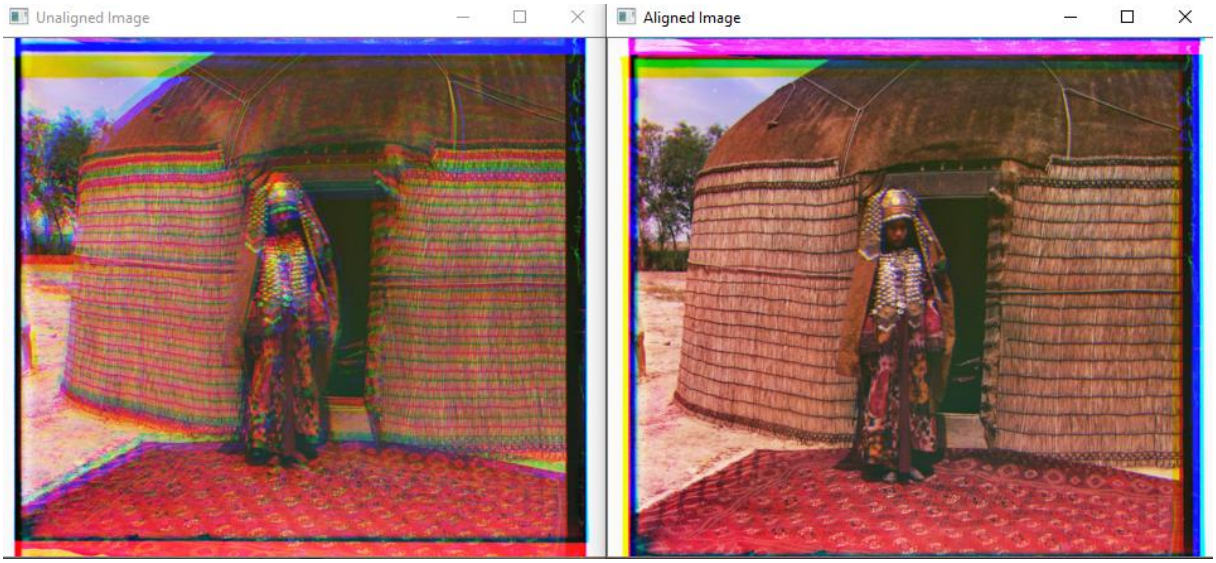
The output above is the total NCC of the range [-20,-20] to [12,4] of the 00029u.jpg image. Range [12,4] is highest sum.

- After that, I found the best NCC with a simple if statement.
- I assigned to "align" the coordinates where I got the best NCC sum and I moved color1 along the y axis according to these coordinates [3].
- I have defined a function named Final_Image(blue,green,red,distance). This function creates the colorized image by calling NCC_Algorithm function in it. Create NCC_Algorithm(red,blue,distance) and NCC_Algorithm(green,blue,distance). Then, "blue", "green_to_blue" and "red_to_blue" are combined respectively with the np.dstack function.

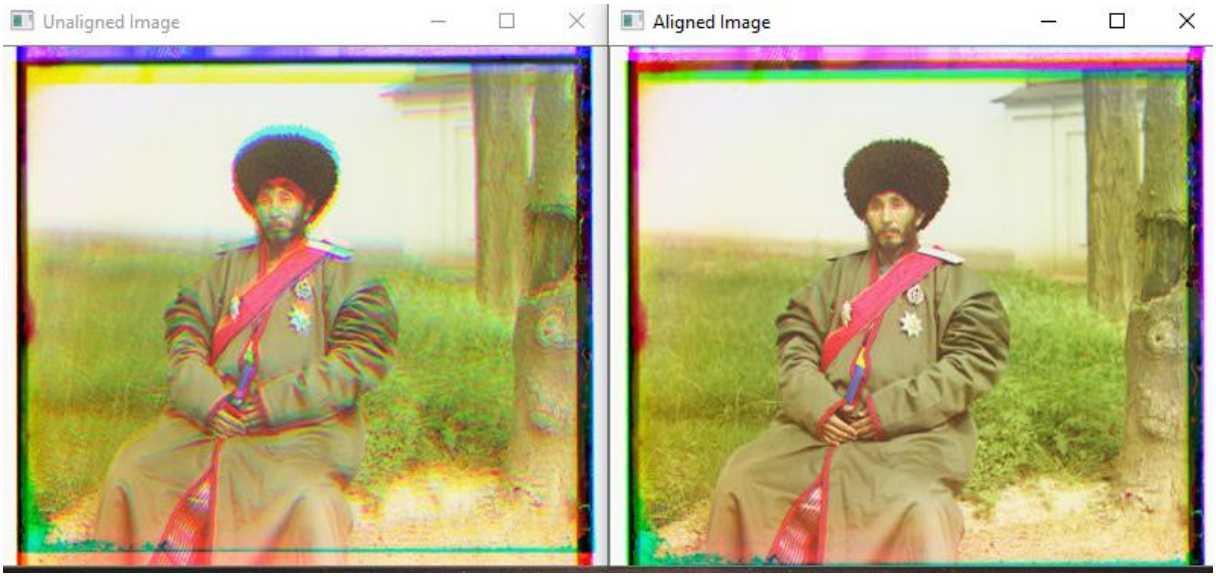
Unaligned Result vs Aligned Result Image 1:



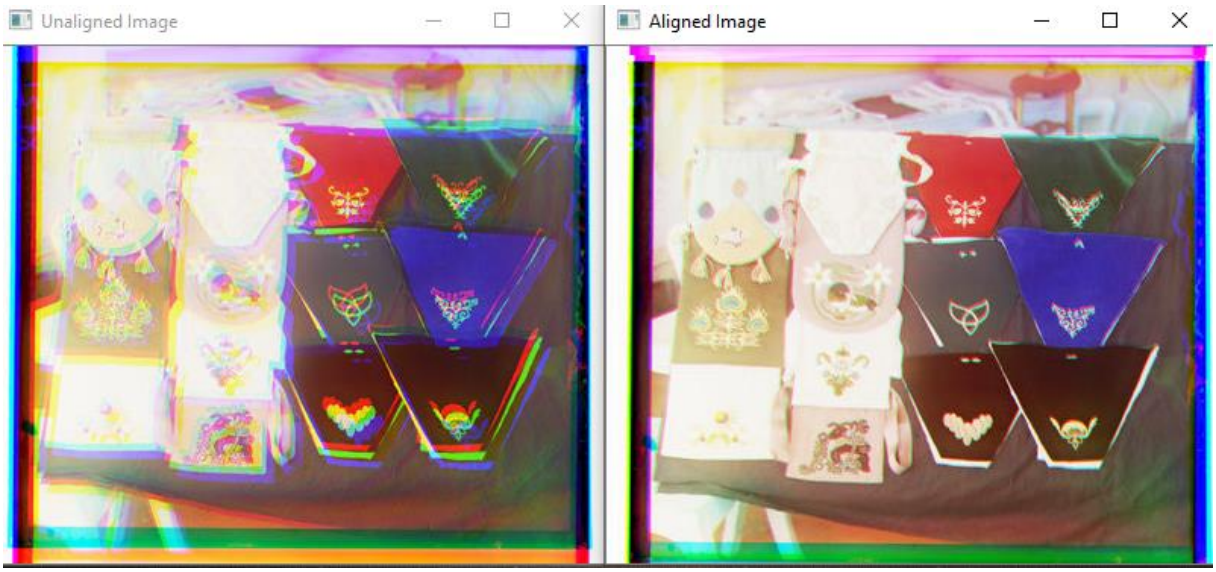
Unaligned Result vs Aligned Result Image 2:



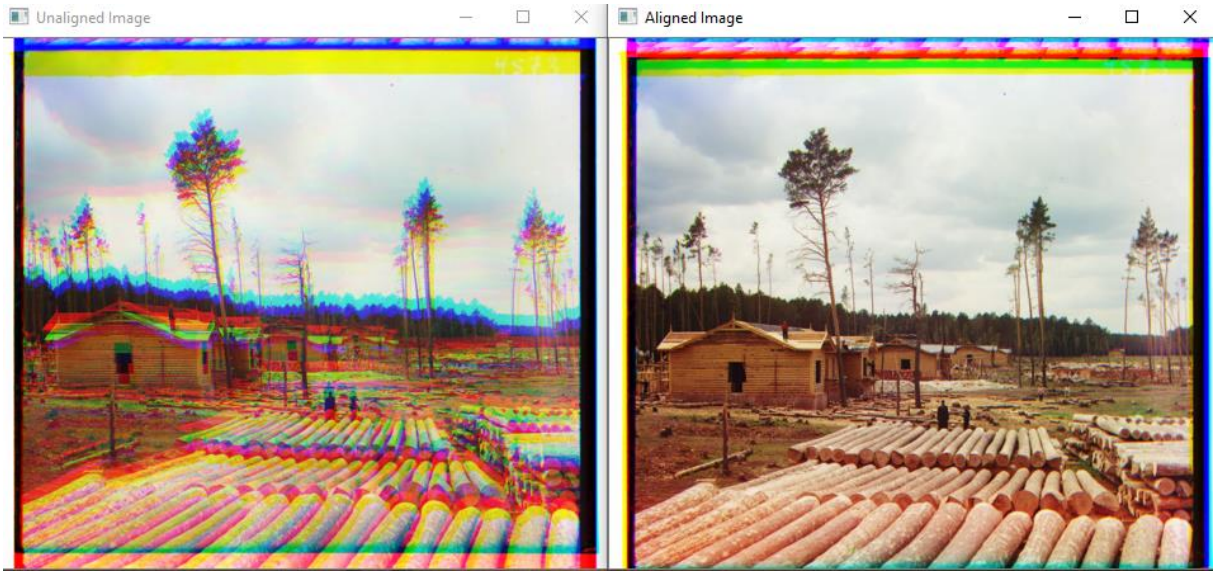
Unaligned Result vs Aligned Result Image 3:



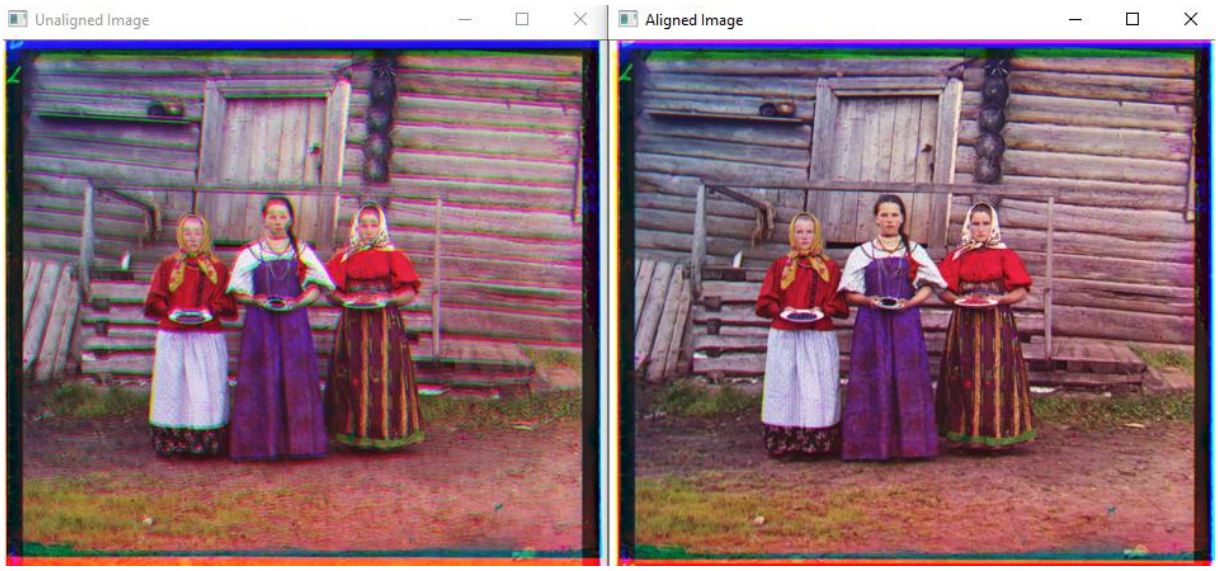
Unaligned Result vs Aligned Result Image 4:



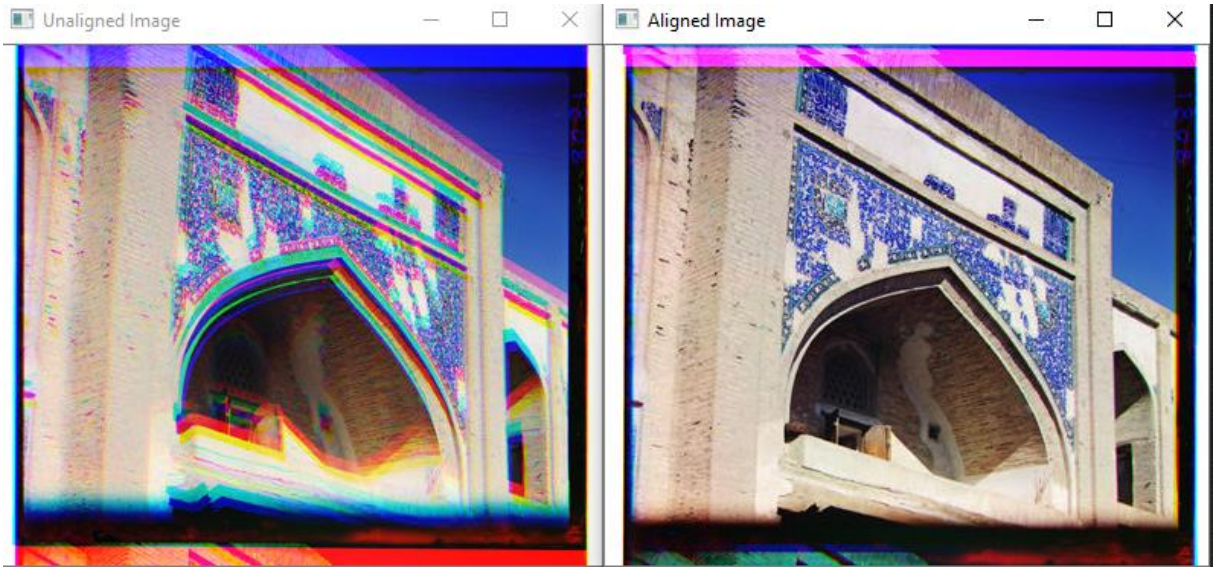
Unaligned Result vs Aligned Result Image 5:



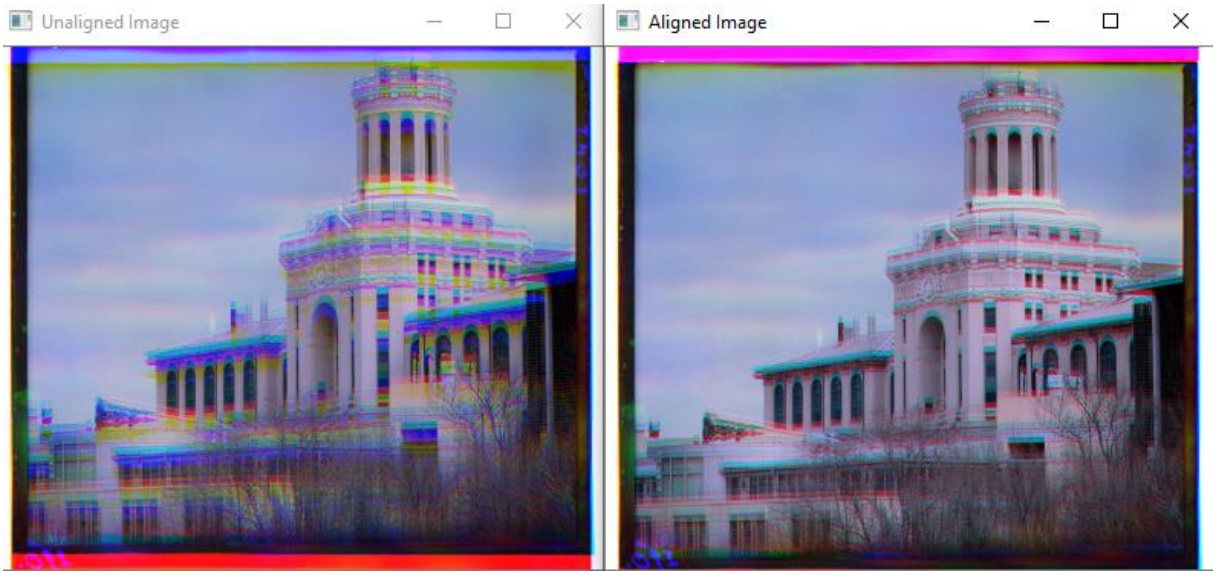
Unaligned Result vs Aligned Result Image 6:



Unaligned Result vs Aligned Result Image 7:



Unaligned Result vs Aligned Result Image 8:



REFERENCES

1. Colorizing the Prokudin-Gorskii Photo Collection. (n.d.). Retrieved from https://inst.eecs.berkeley.edu/~cs194-26/fa17/upload/files/proj1/cs194-26-acg/

2. Images of the Russian Empire:. (2007). Retrieved from http://www.cs.cmu.edu/afs/andrew/scs/cs/15-463/f07/proj1/www/wwedler/

3. Colorizing the Prokudin-Gorskii photo collection. (September,2018). Retrieved from https://andrewdcampbell.github.io/colorizing-the-prokudin-gorskii-photo-collection