CMPE362 HW1

Colorizing the Prokudin-Gorskii photo collection

Eren Kılıç

Department of Computer Engineering

TED University

ABSTRACT

Sergei Mikhailovich Prokudin-Gorskii (1863-1944) was a Russian chemist and photographer. He is best known for his pioneering work in color photography and his effort to document the last years of the Russian Empire. With Tzar's special permission, he traveled across the Russian Empire and took color photographs of everything he saw: people, buildings, landscapes, railroads, bridges, and so on. His idea was simple: record three exposures of every scene onto a glass plate using a red, a green, and a blue filter. His RGB glass plate negatives survived and were purchased in 1948 by the Library of Congress (LoC). The LoC has recently digitized the negatives and made them available online. The glass plate inputs can be converted to a well-aligned and enhance RGB image.

I- INTRODUCTION

In this assignment, we are asked to colorize the given glass plate images. The given inputs contain three different photos and they are merged in a single image. Each photo was taken with different filters. Blue, green and red respectively from top to down. To complete this homework using image processing techniques were necessary for alignment and improvement.

II- STEPS FOLLOWED

A. Separating and Colorizing Image

Initially, I split the image equally into 3 channels and created 3 different channels R, G, B. Then I put G and B channel on top of B since B is my reference point. With these channels together, there was an RGB image now. However, the alignment of the channels was quite bizarre and the image was annoying.

Figure 1: *Image is not proper, needs alignment*

B. Cropping Borders

To get a better result, I tried to ignore borders so created a function that crops border in a specific ration. Mostly 10% is used. Cropping also speeds up aligment process.

Figure 2: *Borderless but still needs alignment*

C. Aligning Channels

I used two different alignment methods.

1-) SSD

The first method called SSD. To align three channels, the program moves the second two channels -G and R, by [-15,15] pixels in both x and y axes in a brute-force way, and calculate the Sum of Squared Differences of two channels. After the computation, pick the best one and apply the offset to the G and R channels. After all, I merged the channels with MATLAB's cat function.

Figure 3: *Image is aligned with SSD*

2-) NCC

NCC is the abbreviation of normalized cross-correlation. It is very similar to SSD but it uses NCC instead of SSD to find the best offset. To align the image I used MATLAB's normxcorr2 function.

Figure 4: *Image is aligned with NCC*

3-) NCC vs SSD

Even though, at first glance at least for that image "lady.png", both methods seem like performing the same the result may differ based on the input image. I will mention this subject later on.

D. Improving the Quality

My program has different approaches to enhance the quality of the already aligned image.

1-) Gamma Correction

In MATLAB, imadjust function can be used to adjust the contrast of an image by specifying a gamma value. I used 1.5 as a gamma value to a get more vivid image.

Figure 5: *Gamma correction is used (gamma val = 1.5)*

2-) Histogram Equalization

Likewise gamma correction, histogram equalization strategy can be used to improve contrast as well. In MATLAB the function is called histeq.

Figure 6: *Histogram equalization is used*

3-) Working on HSV Color Space

HSV stands for hue, saturation and value. In MATLAB users can convert an RGB image to HSV image (or vice versa) with rgb2hsv function. I used this function to change the contrast of value. In order to this, first I split the HSV image into 3 channels and then applied histeq function to “value”. Now, I had a new value arranged with histogram equalization. By using this value I created a new HSV image with cat function. Finally, I converted the HSV image to RGB image with hsv2rgb function.

Figure 7: *Adjusted HSV’s v value with histeq*

III-COMPARISONS

For the image above “lady.png”, both SSD and NCC areperforming well. Yet, these may not be valid for all images such as “self_portrait.png” or “emir.png”.

Figure 8: *Not aligned emir.png with borders*

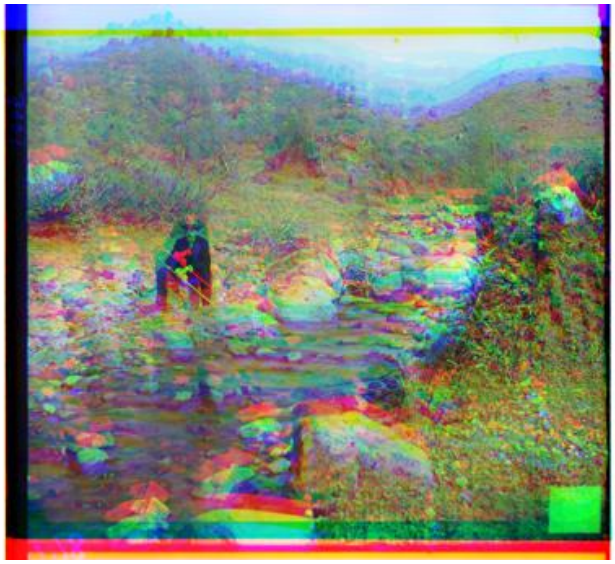

Figure 9: *Not aligned self_portrait.png with borders*

1-) Appliying SSD Alignment

Figure 10: *Borderless, SSD aligned emir.png*

Figure 11: *Borderless, SSD aligned self_portrait.png*

Obviously, SSD alignment on “emir.png” is is successful as shown in Figure 10 whereas the result is not the same for self_portrait.png. The man’s face in the river is not clear.

2-) Appliying NCC Alignment

Figure 12: *Borderless, NCC aligned emir.png*

Figure 13: *Borderless, NCC aligned self_portrait.png*

In Figure 12, we see that NCC algorithm did not work well contary to SSD. However, this time “self_portrait.png” is very well aligned with NCC method and the man’s face is pretty clear.

IV. SUMMARY

In conclusion, it is clear that SSD and NCC are mostly successful for small images. They make alignment in several seconds but the elapsed time might be larger if the size of input image grows. Hence, we should try different approaches like pyramid aligment. In my program I also used cropping to speed up. Additionaly, aligned images can be improved with various implementations like histogram equalization, gamma correction or working on hsv to alter value by histeq function. Furthermore, in my program I crop borders 10%. Unexpectedly, for the “three_generations.png” image NCC alignment did not work but when I set the cropping ratio as 11.9%, NCC works very well.

V. REFERENCES

[1] The Prokudin-Gorskii Photographic Record Recreated: The Empire That Was Russia. (n.d.). Retrieved from http://www.loc.gov/exhibits/empire/gorskii.html