Assignment 3

Images of the Russian Empire

Erik Zorn-Wallentin
0864583
CIS*4720 - Image Processing

I. Introduction

Assignment 3 for CIS*4720 gave us several tasks to choose from and complete using image processing techniques recommended by our professor and course notes. The task I chose was “Images of the Russian Empire”, which is based on a man called Sergei Mikhailovich Prokudin-Gorskii (1863-1944) and he created grayscale images that represent three filtered images (R-G-B) and it would be combined to create a composed colour image.

I did this assignment by myself. I was able to create a successful algorithm that would create a coloured image based on RGB filter images.

II. Prokudin-Gorskii Images

There is a collection of Prokudin-Gorskii images on the internet that can easily be downloaded from [1].

An algorithm was needed to register Prokudin-Gorskii images which could be attempted many different ways. One way could be to ignore small nonlinear difference, and another way to deal with the nonlinear differences. There were also some assumptions of these images where the camera is stationary, there is a time-gap between the images of around 1-3 minutes, small affine changes in the scenes, and small nonlinear changes like movement of clouds or trees, etc.

If you just get the grayscale RGB images and simply split them into the 3 R, G, B images and try to merge for the coloured image the result will have problems, as it doesn’t take into consideration of the linear changes, nonlinear differences, or any problems between the 3 images. The goal is to take into all of the problems listed above, and still create a coloured image from the RGB filtered images and fix all of the problems if there are any with minimal visible artifacts.

There was some tips and recommendations provided in this assignment, where the professor recommended to solve it from intensity based registration, or control selection and matching by means of SIFT. I did research in trying to find different techniques to do this task and tried many different approaches.

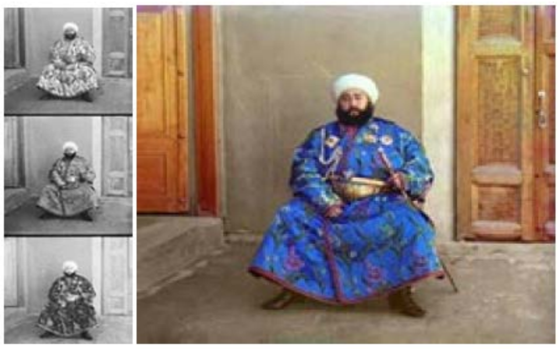
Figure 1 – Prokudin-Gorskii Image [2]

III. Process

There was a specific process I took to complete this task. The photo collection provided has a single image which is actually 3 separate images in R, G, B channels. So the first task is I need to split the 3 images into the R, G, B channel. Since the way they were taken there is some background border around the images, and this is not useful so there is a crop applied to it before image is split into the 3 channels.

To try and deal with the problems listed above like the nonlinear differences in the images, or the recommendations by the professor I implemented a normalized cross-correlation algorithm. Now the algorithm itself is a decently heavy computational task which is described in more detail below, and another problem is the image pixel density is a decently large size (3000x3000 pixels). To make the process algorithm calculation process quicker the image is rescaled to a smaller size, before doing the NCC algorithm.

When you provided an image, it will convert it to the 3 split R, G, B channels and use the NCC algorithm and merge the 3 channels after to create a coloured image. I present the 3 R, G, B images to the user in the directory and the result coloured image. You can also see other results in the “result” directory.

IV. Normalized cross-correlation

The algorithm I implemented to complete this task is called normalized cross-correlation (NCC), which is an extension of CC. The CC is a normalization by dividing it by the product of the root-mean squared intensities of each of the two images [2]. It is an invariant to any linear intensity transformation.

$$NCC(i,j) = \frac{\sum_{x=-Mx}^{Mx} \sum_{y=-My}^{My} S(x+(Mx+1), y+(My+1)) R(i+x, j+y)}{\left[\sum_{x=-Mx}^{Mx} \sum_{y=-My}^{My} S(x+(Mx+1), y+(My+1))^2\right]^{0.5} \left[\sum_{x=-Mx}^{Mx} \sum_{y=-My}^{My} R(i+x, j+y)^2\right]^{0.5}}$$

Figure 2 – NCC Formula [2].

The pseudo code for the algorithm is it checks the source image and the second image provided, for example the G and B channel where the G is the source. Now it will do a cross-correlation by the group and will be normalized between -1 and 1 and in my case also a pixel amount range added. The image would be displayed by a certain x and y pixel amounts in the horizontal / vertical direction. Finally it will check for the max alignment amount and return that.

V. Tests and Results

There is 6 test images from the Prokudin-Gorskii photo collection provided in the assignment directory. I applied my algorithms to the images, such as the NCC algorithm as described above.

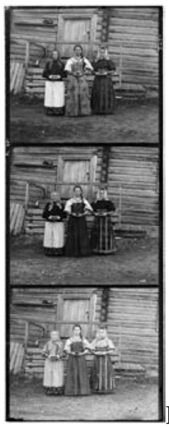
Figure 3 – 1.png

Using the first algorithm, which is very simple it just extracts the 3 channel image from the image in Figure 3, after it will simply merge the 3 channels and create the Figure 4 result.

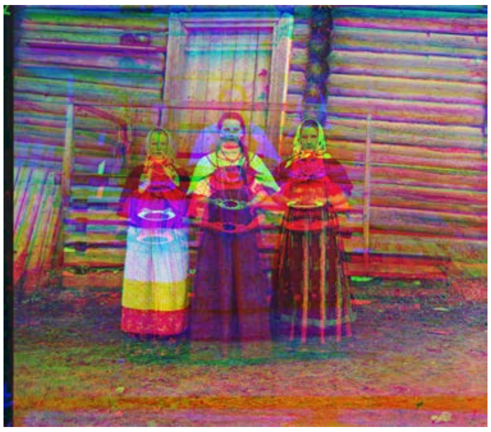

Figure 4 – First algorithm merged 1.png

Looking above in figure 4, you can clearly this just merging the 3 R, G, B channels and not taking into consideration and affine changes or nonlinear changes will give this result, which is incorrect.

Now following using the second algorithm, the NCC described above it would result in the image below in Figure 5.

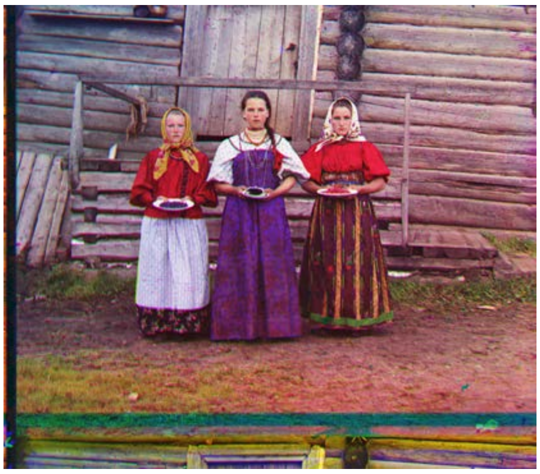

Figure 5 – Result from 1.png using NCC

Testing this image, I was successfully able to extract the 3 channel images (R, G, B) and create the coloured image on it while fixing the major problems in the image. There is some artifacts remaining in the image which I will explain more later.

Comparing Figure 5 algorithm to Figure 4 algorithm, you can clearly see a difference in the result, where the NCC algorithm clearly fixes the image and addresses the problems that were discussed in the introduction.

More results of tests I done on all 6 images are shown in the result directory, please see that directory as it was tested on 6 images, which are not all shown in this paper.

VI. Artifacts and Improvements

With more time I would have tested on more images, created more algorithms to fix the problems that my current algorithms couldn’t fix. In figure 5 you can see some weird artifacts around the eyes, and giant green line at the bottom.

For example the NCC algorithm could not handle some images well at all, in Figure 6 below you can see a lot of blurriness in this image, and a lot of artifacts all around, so in this case the NCC algorithm was not very successful, where other algorithms may do a better job. NCC algorithm also has a heavy computation load, and takes a long time on bigger images.

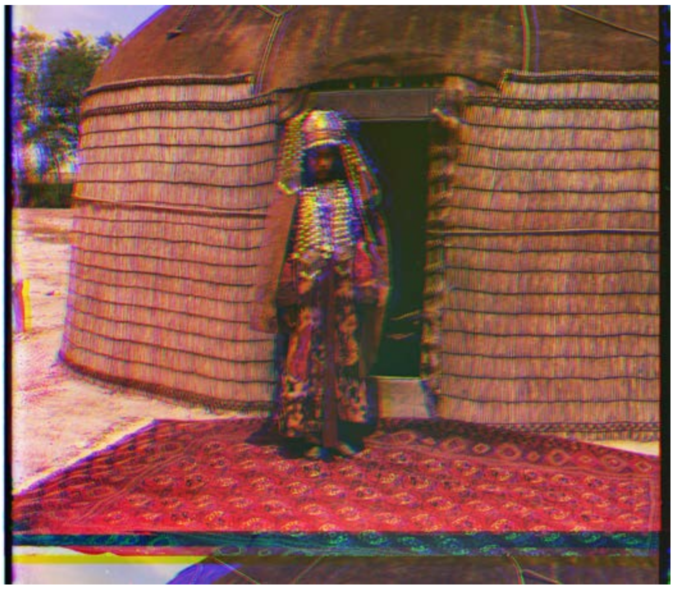

Figure 6 – NCC Result of 3.png

Under more time I would attempt to fix some of these problems, or try to take into consideration of these problems and work an algorithm that would address it.

References

[1] Prokudin-Gorskii Collection – About this Collection http://www.loc.gov/pictures/collection/prok/

[2] CourseLink Retrieved from https://courselink.uoguelph.ca/d2l/home/454932

[3] Getting Started with Images Retrieved from http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_image_display_py/py_image_display.html

[4] OpenCV: Geometric Transformations of Images Retrieved from http://docs.opencv.org/trunk/da/d6e/tutorial_py_geometric_transformations.html

[5] The Image Module Retrieved from http://effbot.org/imagingbook/image.htm

[6] Computing cross-correlation function Retrieved from http://stackoverflow.com/questions/6991471/computing-cross-correlation-function