CS543/ECE549 Assignment 1

Name: Yiyang Liu
NetId: yiyang34

Part 1 : Implementation Description

Provide a brief description of your implemented solution, focusing especially on the more "non-trivial" or interesting parts of the solution.
What implementation choices did you make, and how did they affect **the** quality of the result and the speed of computation?

- What are some artifacts and/or limitations of your implementation, and what are possible reasons for them?

Basic alignment.

I use exhaustive search to align the low-resolution images. In the exhaustive search, I move two of three channels pixel to pixel in a range of a [-15, 15] window to find the place with the best score, which, to be specific, is normalized cross-correlation (NCC) in my implementation. However, the result seems badly after this process. One of the reasons that cause this poor quality is the dark border on every original image. To get rid of the negative impact of the dark border, I crop out the dark border of the original images during the preprocessing period. Therefore, improve the accuracy of the loss evaluation and image alignment.

Multiscale alignment.

A coarse-to-fine search method is applied in my implementation. This method first creates an image pyramid using Gaussian down sampling and then starts to align image from the coarsest to the finest image in the pyramid. In this way, the running time to process the image is reduced in a significant extent.

I also slightly expand the window size, which is the pixel inspection region mentioned above, to avoid unexpected error in the NCC evaluation function. The common used strategy is by looking through only four pixels in each pyramid image while I use nine pixels, which is a three times three matric. This adjustment sacrifices running time to makes the algorithm more robust to local minima.

The issue of dark border still remains, because it is unrealistic to crop all the image with same coordinates. In future work, we can design an algorithm to crop off out the dark border automatically.

Part 2: Basic Alignment Outputs

For each of the 6 images, include channel offsets and output images. Replace <C1>, <C2>, <C3> appropriately with B, G, R depending on which you use as the base channel.

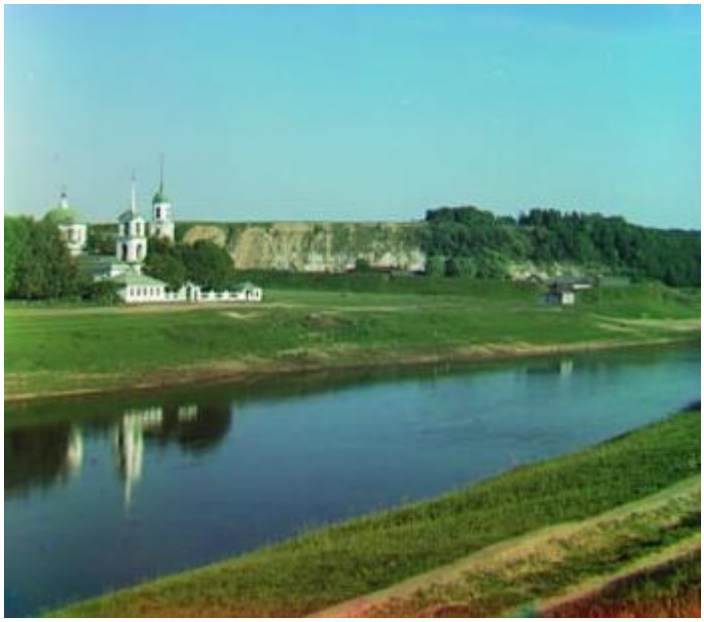
A: Channel Offsets

Using channel <B> as base channel:

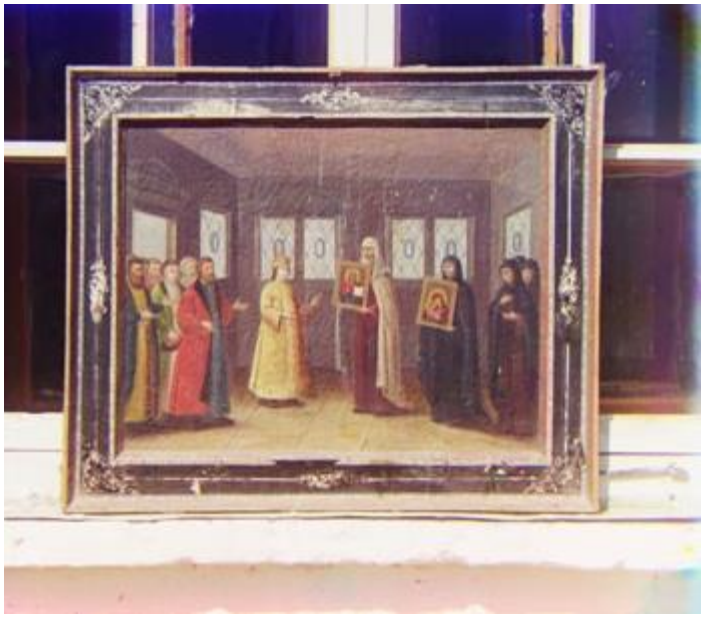
Image	<G> (h,w) offset	<R> (h,w) offset
00125v.jpg	(2, -8)	(1, -4)
00149v.jpg	(2, -9)	(2, -5)
00153v.jpg	(3, -6)	(5, 0)
00351v.jpg	(1, -9)	(1, -1)
00398v.jpg	(3, -8)	(4, -3)
01112v.jpg	(0, -13)	(1, -9)

B: Output Images

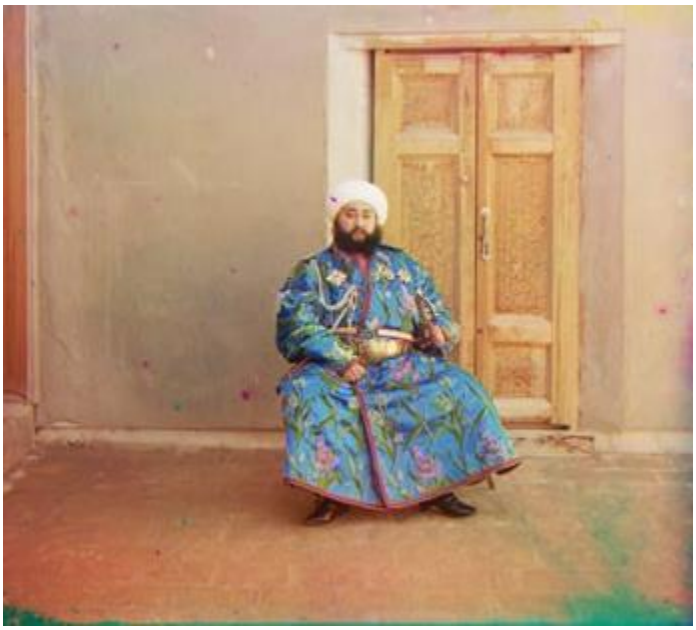
Insert the aligned colorized outputs for each image below (in compressed jpeg format):



00125v.jpg



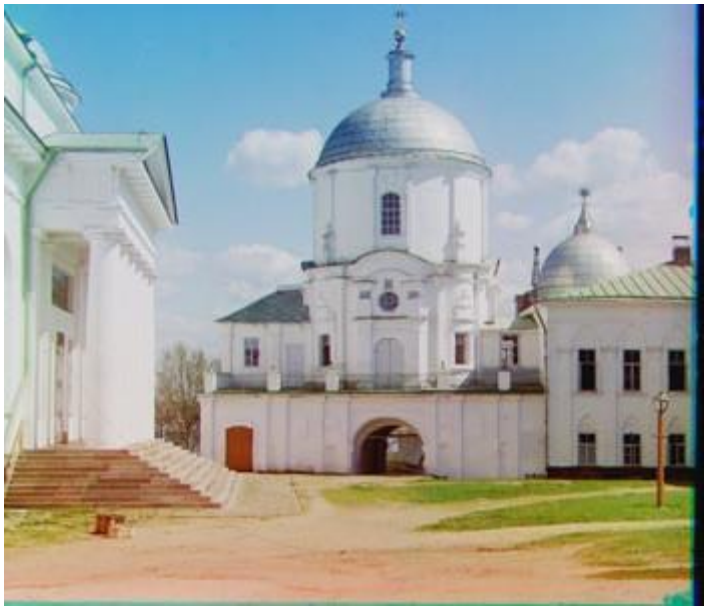
00149v.jpg



00153v.jpg

00351v.jpg

00398v.jpg



01112v.jpg

Part 3: Multiscale Alignment Outputs

For each of the 3 high resolution images, include channel offsets and output images. Replace <C1>, <C2>, <C3> appropriately with B, G, R depending on which you use as the base channel. You will also need to provide an estimate of running time improvement using this solution.

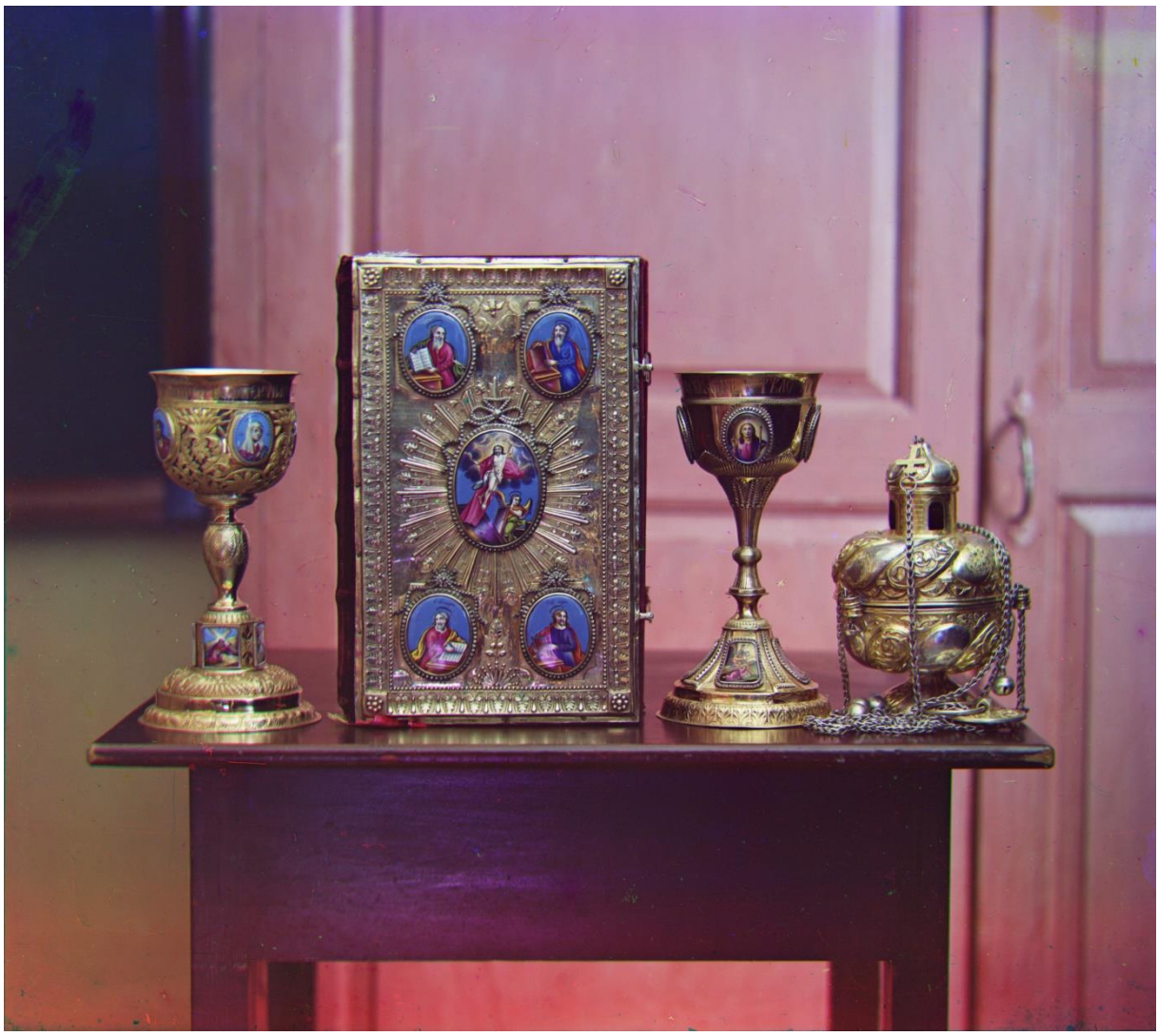
A: Channel Offsets

Using channel <B> as base channel:

Image	<G> (h,w) offset	<R> (h,w) offset
01047u.tif	(24, -82)	(29, -28)
01657u.tif	(18, -32)	(25, 31)
01861a.tif	(39, -80)	(41, -35)

B: Output Images

Insert the aligned colorized outputs for each image below (in compressed jpeg format):



01047u.tif

01657u.tif

01861a.tif

C: Multiscale Running Time improvement

Report improvement for the multiscale solution in terms of running time (feel free to use an estimate if the single-scale solution takes too long to run). For timing, you can use the python `time` module, as described in the assignment instructions.

I compare the running using `01861a.tif` as example image before and after applying multiscale solution. For single-scale solution, I apply a 90×90 pixel size window, because the biggest absolute offset value is 80. And for multiscale solution, I use an eight layers' image pyramid. Undoubtedly, the multiscale solution runs much faster (×1000). More importantly, the multiscale solution is less likely to get crapped in the local minima.

Image	single-scale solution running time	multiscale solution running time
01861a.tif	9720s	11.64s

Part 4 : Bonus Improvements

Post any extra credit details with outputs here.