CS543/ECE549 Assignment 1

Name: Wei Chieh Huang
NetId: whuang73

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?
    - I chose the SSD method for finding my offset/displacement. It was the first suggestion on finding displacement from the assignment description document so I went with it. Originally, I feel like I was getting the wrong offset from my algorithm code and I couldn't find where and what is causing this problem. Where I decide to use the NCC approach. NCC approach takes longer to run and I was getting a similar result to SSD. I finally figure that I need to take the opposite value of the offset and the resulting image become what I expected it to be. In the end, I have both approaches working really great. Finally, I decide to go with the SSD method since the run time is much faster than NCC in my case.
    - I also implement the image pyramid method for the multiscale alignment. It takes longer to process than I expected at first when I keep the window size the same. After I reduced the window size based on the number of recursions. It went so much faster.

- What are some artifacts and/or limitations of your implementation, and what are possible reasons for them?
    - If we wanted to be more precise and increase the window size. The running time will increase because it will need to compare more areas.
    - One of the artifacts was the output of 00153v.jpg which doesn't go quite as expected. It seems like one of the layers does not match precisely. I believe the possible reason might be the size of the search window. I did some experiments on increasing the size of the window but it only improve a little on precision and takes a longer running time. As a result, I decide to have a window size that works for the majority of the cases.

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.

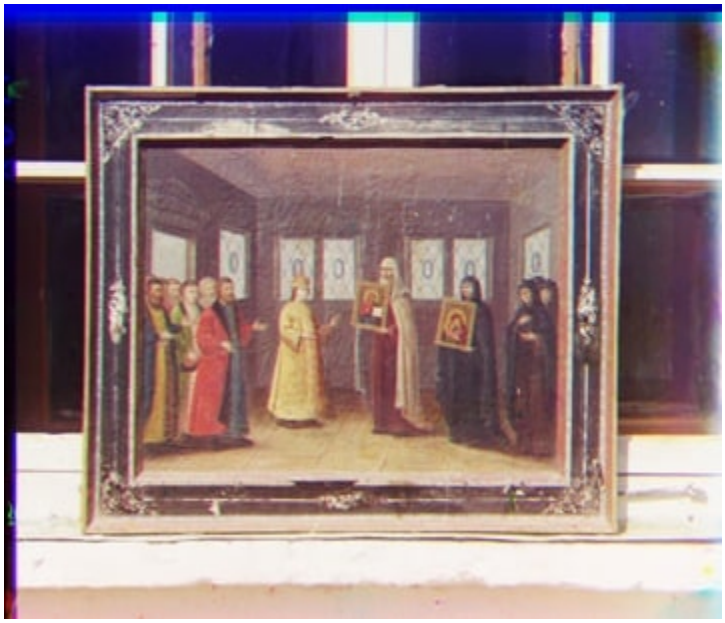
A: Channel Offsets

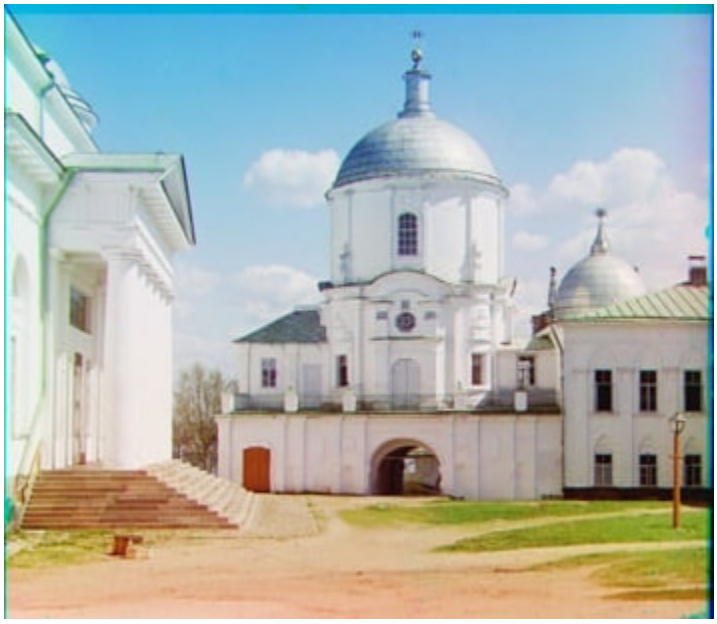
Using channel <B> as base channel:

Image	<G> (h,w) offset	<R> (h,w) offset
00125v.jpg	[1, 6]	[1, 10]
00149v.jpg	[2, 4]	[1, 9]
00153v.jpg	[2, 7]	[0, 2]
00351v.jpg	[0, 4]	[1, 13]
00398v.jpg	[0, 5]	[0, 12]
01112v.jpg	[0, 0]	[1, 5]

B: Output Images

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





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.

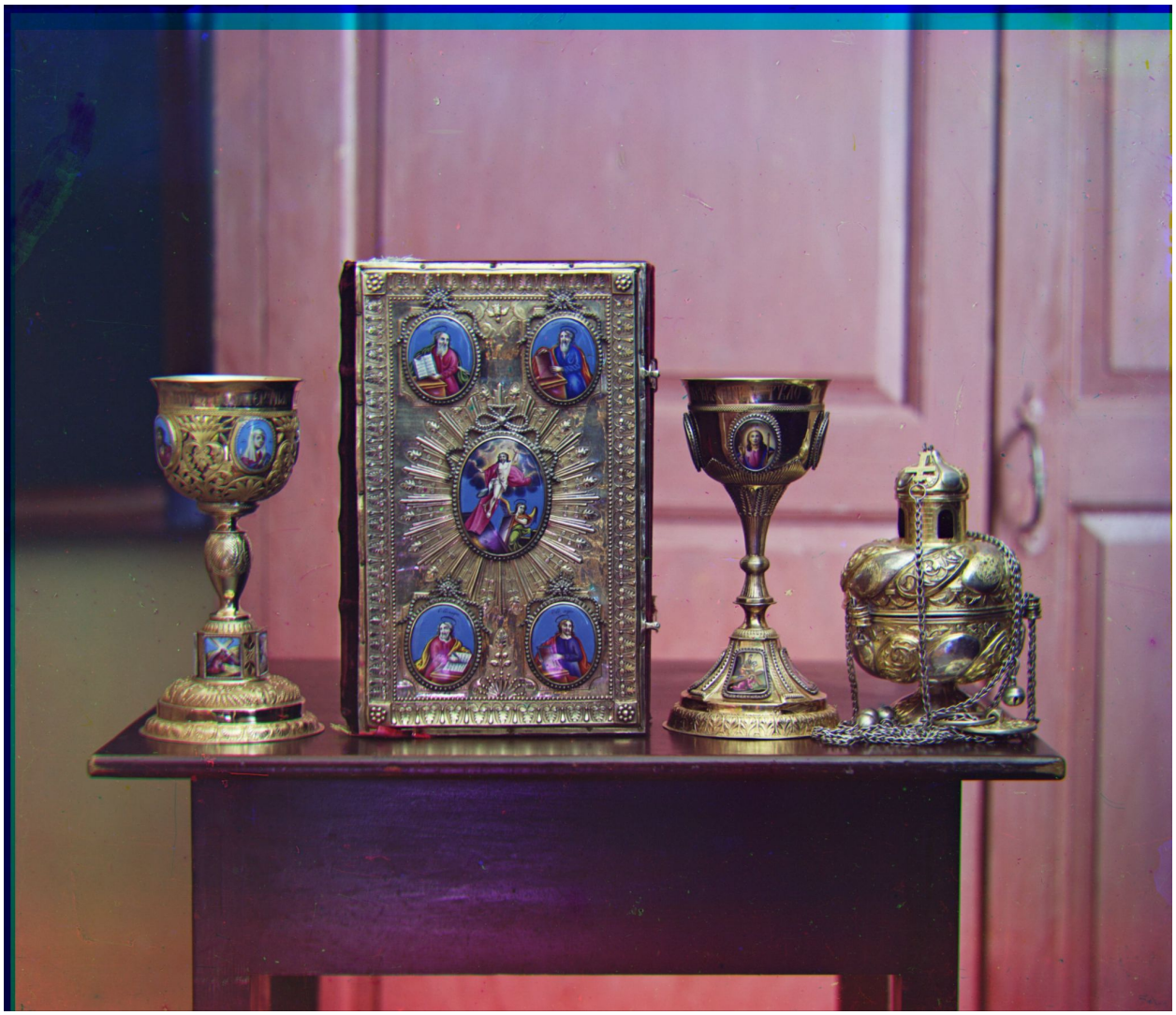
A: Channel Offsets

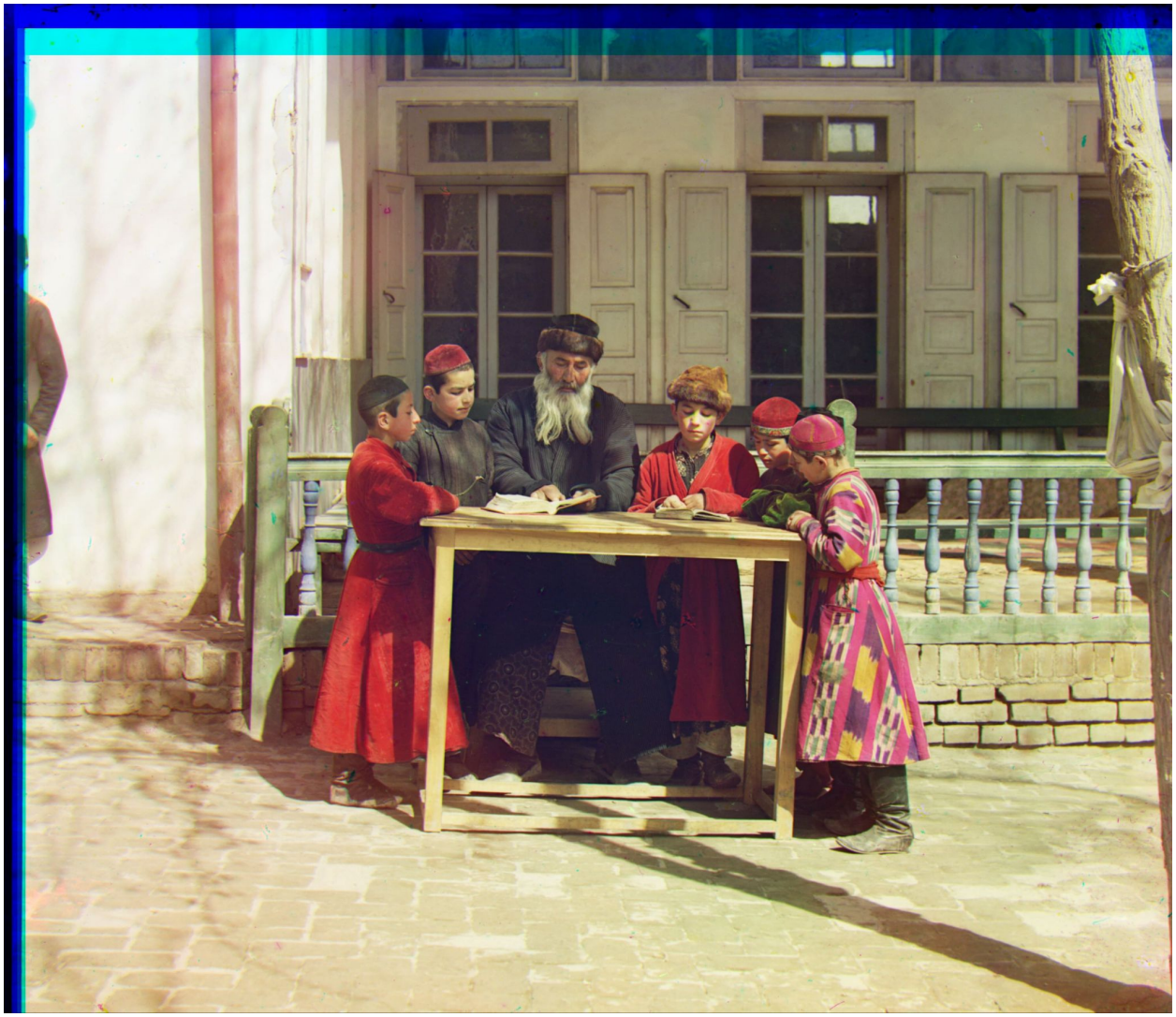
Using channel <B> as base channel:

Image	<G> (h,w) offset	<R> (h,w) offset
01047u.tif	[20, 24]	[33, 72]
01657u.tif	[8, 53]	[10, 117]
01861a.tif	[39, 71]	[63, 148]

B: Output Images

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





C: Multiscale Running Time improvement

Time before improvement:

Time estimate: 9 minutes

since it takes too long to run with only SSD method. I estimated the time by using the result of the basic alignment which normally takes around 5 seconds for each image. The estimation is simply comparing the shape and as a result, it is about 100 times larger than the image of basic alignment.

Time after improvement:

Time: ~15 seconds

After using the image pyramid method. The running time was reduced tremendously. Each case has a runtime of around 15 seconds. Although it could be a little faster than this by

changing the window size of the image rolling or lowering more resolution, however, it would lose some precision and resolution. Therefore, I have a window size that feels the most reasonable to me as it has decent precision and running time.

Part 4 : Bonus Improvements

Post any extra credit details with outputs here.

The additional idea for me was trying to get a better or faster result using the NCC method as I mentioned earlier. The result I'm getting is the same but it takes ~24 seconds to run the method. When compares to the average time of SSD, which is ~5 seconds, NCC takes 4 times longer on running time. For this reason, I decide to choose SSD as the method of processing images.

```
PS C:\CS543> & C:/Users/josep/AppData/Local/Programs/Python/Python310/python.exe c:/CS543/second.py
True
[-1, -6]
[-1, -10]
23.991676568984985
```