CS543/ECE549 Assignment 1

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 cut original image into three pieces, and fix the first one, choose a 15*15 window in the center of the first image, and slide the second and third images upon the first one to calculate offsets. I first slide across the whole picture, then I tried to slide only within the center, which also generates a good result with less time.

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

- Two out of six samples have strange offset regarding C3. After trying to enlarge window size or limit the searching area from all across the image to center only, I successfully generate a better image for 00351v.jpg.

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

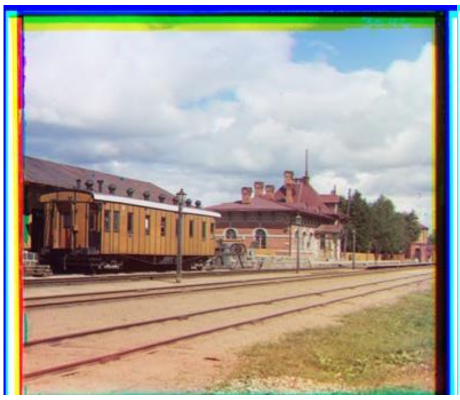
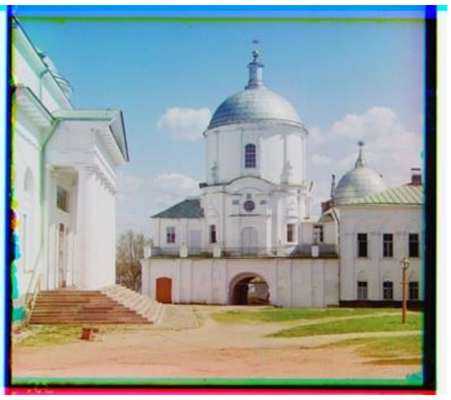
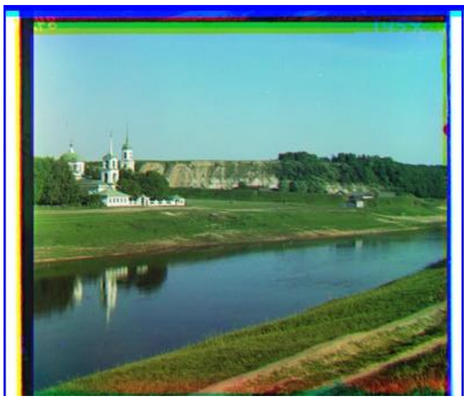
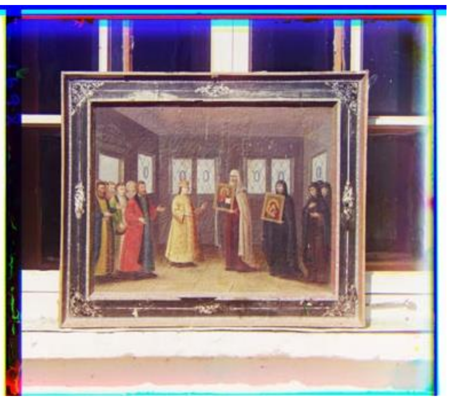
Using channel <C1> as base channel:

Image	<C2> (h,w) offset	<C3> (h,w) offset
00125v.jpg	(5,2)	(10,2)
00149v.jpg	(4,2)	(9,2)

00153v.jpg	(7,3)	(-126,-85) (15,-10)(after changing limitations)
00351v.jpg	(4,1)	(-138,-154)(13,1)(after changing limitations)
00398v.jpg	(5,2)	(11,4)
01112v.jpg	(0,0)	(5,1)

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.

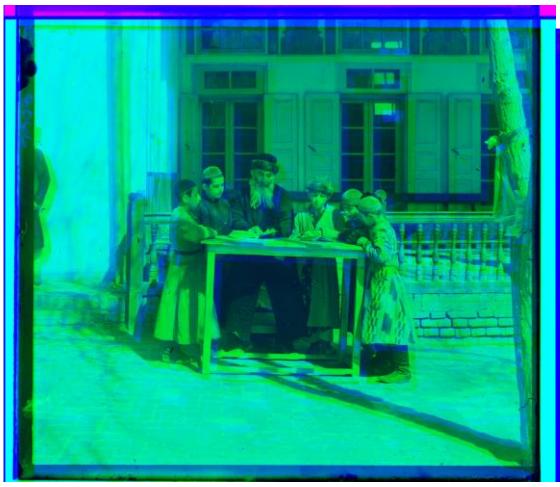
A: Channel Offsets

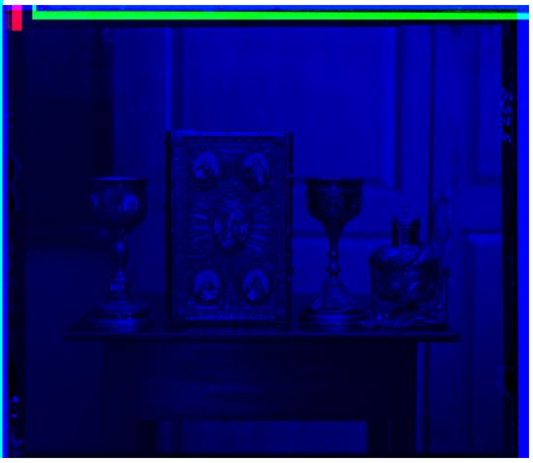
Using channel <C1> as base channel:

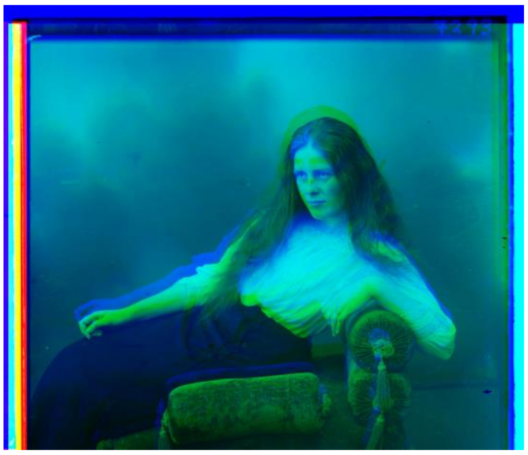
Image	<C2> (h,w) offset	<C3> (h,w) offset
01047u.tif	(-182,-137)	(-103,212)
01657u.tif	(134,35)	(120,-168)
01861a.tif	(-74,5)	(98,12)

B: Output Images

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







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.

Image	Original time(estimated)	New time
01047u.tif	10360 seconds	12.48 seconds
01657u.tif	10360 seconds	21.24 seconds
01861a.tif	10360 seconds	12.36 seconds

Part 4 : Bonus Improvements

Post any extra credit details with outputs here.

The image generated from 00351v.jpg and 00153v.jpg shows unpreferable results, so I enlarged the window size to 50 and successfully generate a good image for 00351v.jpg. I also try to change the searching area from all across the map to only search inside the center, which gets the result and simplify the code, as well as consuming less time.