CSC320: Assignment #1

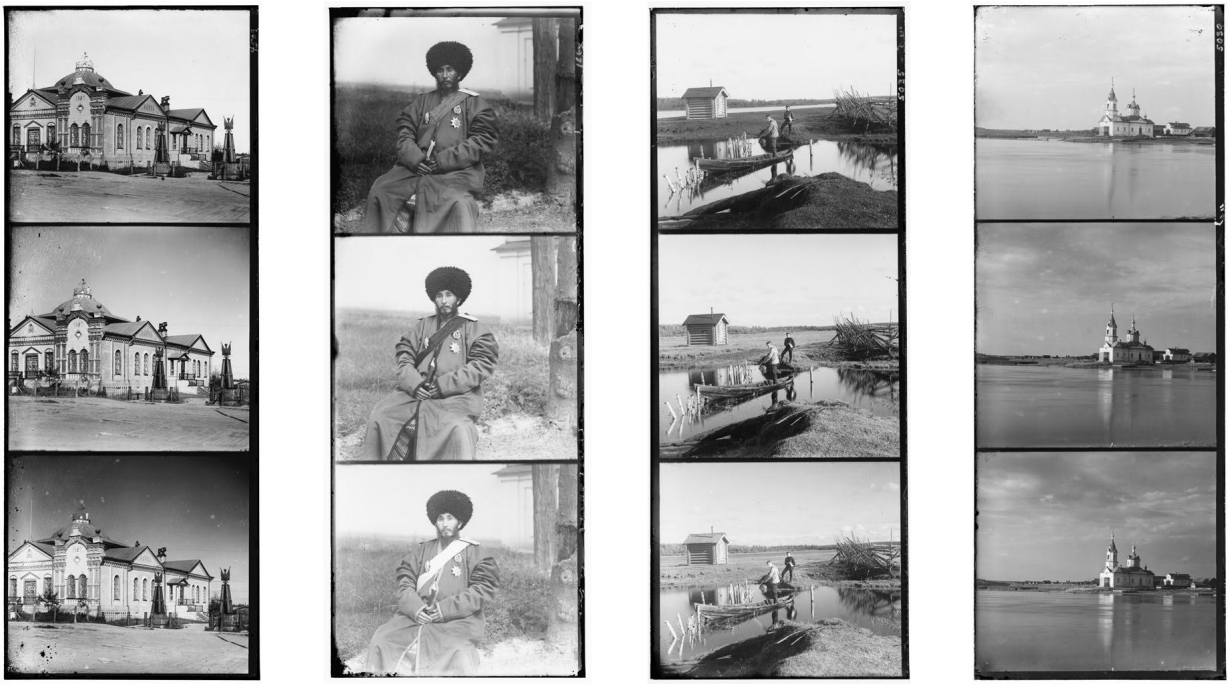
Due on Saturday, January 24, 2015

Sang-Ah Han

January 24, 2015

Part 0

Dataset description
The dataset consists of images that contain 3 vertically stacked photos that have been taken in greyscale, but through blue, green, and red filters respectively. Four examples of these images are shown in Figure 1.



(a) (b) (c) (d)

Figure 1: Low-resolution sample input photos.

To align the photos properly, the black borders around each photo must be taken into account when applying a patch matching method. Some of the input photos are lower-resolution *.jpg* files, while others are higher-resolution *.png* files.

Part 1

Applying an algorithm: initial exploration.
Using the low-resolution *.jpg*'s from the given dataset of images, colour images were obtained using both *Normalized Cross-Correlation* (NCC) and *Sum of Square Differenced* (SSD) template matching methods. For the most part, the two methods, SSD and NCC, produced very similar results, as depicted in Figure 2, and in Figure 3

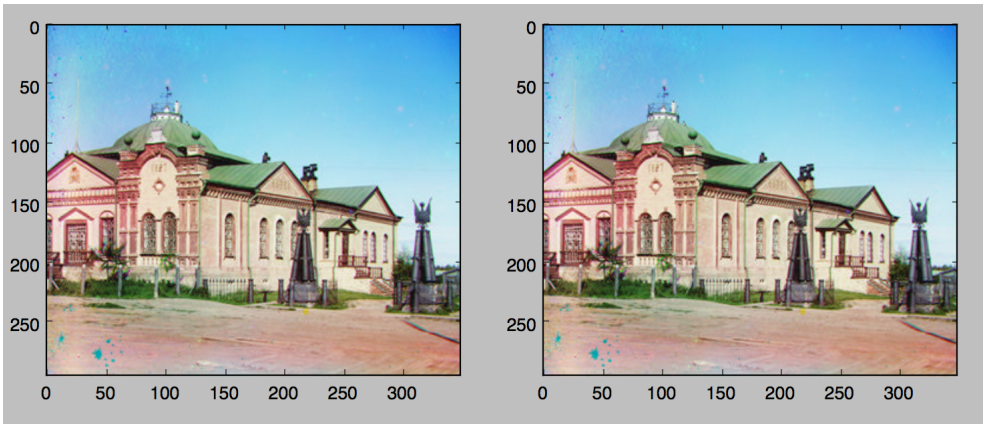


Figure 2: Left: coloured version of Figure 1(a), aligned with SSD. Right: coloured version of 1(a), aligned with NCC.

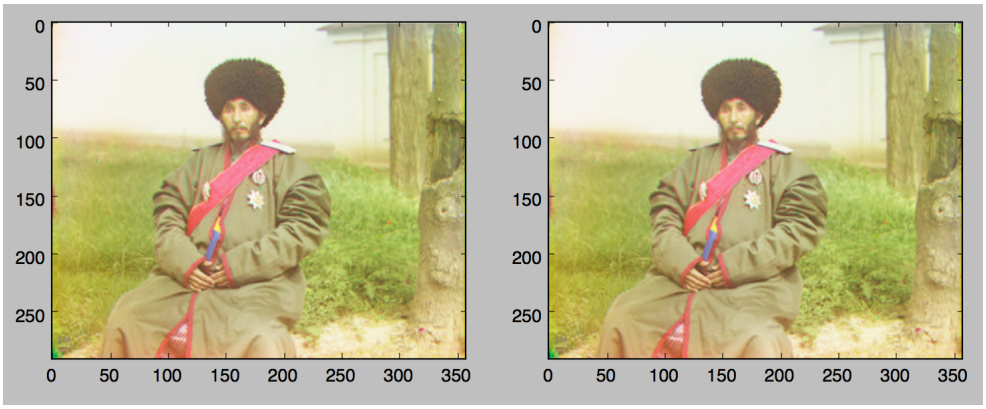
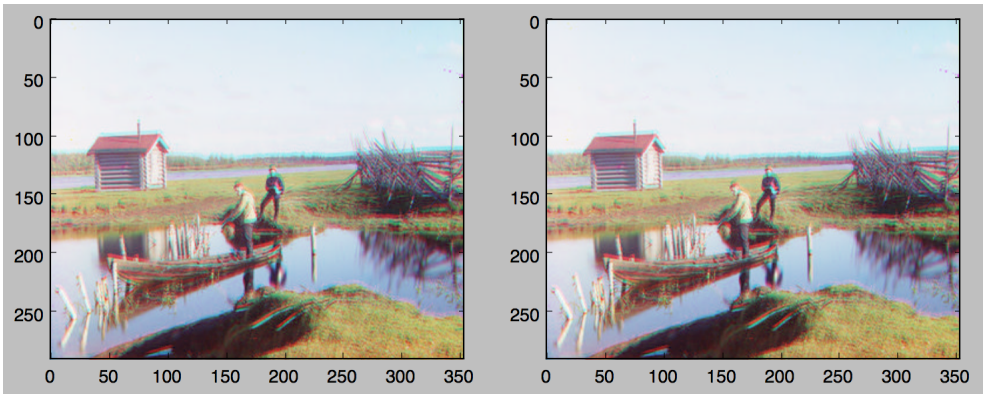


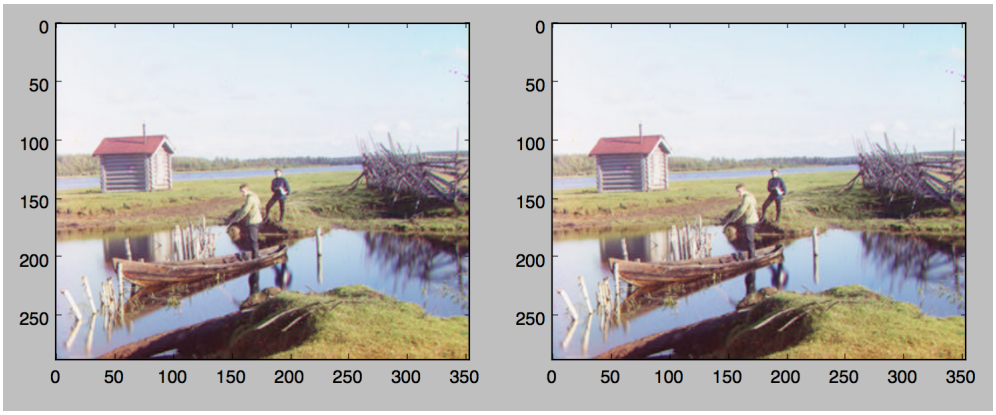
Figure 3: Left: coloured version of Figure 1(b), aligned with SSD. Right: coloured version of 1(b), aligned with NCC.

However, not all of the inputs produced perfect outputs, for various reason. One being the range of displacement used to test alignment of the photos– this task specified to test alignment by displacing the photos 10 pixels each way (left, right, up and down) and obtaining the displacement with the best score. For some images, this score was best achieved with a displacement of more than 10 pixels, as shown in Figure 4.

Part 1 continued on next page...



(a) Left: coloured version of Figure 1(c), aligned with SSD. Right: coloured version of Figure 1(c), aligned with NCC. Both images checked for alignment for displacements up to 10 pixels.



(b) Left: coloured version of Figure 1(c), aligned with SSD. Right: coloured version of Figure 1(c), aligned with NCC. Both images checked for alignment for displacements up to 15 pixels.

Figure 4

Part 1 continued on next page...

The resulting photos from applying both SSD and NCC in Figure 4(a) were not enough to align the images perfectly. It was only until the range of displacements to test alignments was increased (shown in Figure 4(b)) that it was possible to align them perfectly.
Finally, there were some images for which the two methods did not yield similar results. Figure 5 shows an example of this occurrence. Though SSD is more computationally efficient compared to NCC, SSD is also more sensitive to the intensity of the images being compared. NCC avoids this by subtracting the mean and dividing by the standard deviation for every point.

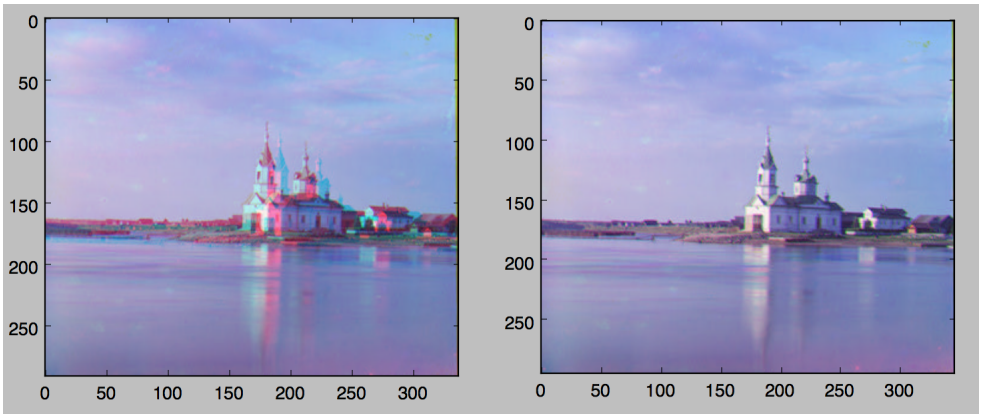


Figure 5: Left: coloured version of Figure 1(b), aligned with SSD. Right: coloured version of 1(b), aligned with NCC.

From the output of these sample inputs, as well as other input images from the dataset, it seems that using NCC to align the photos would result in more well-aligned output photos. Though SSD requires a lot less computation, SSD is more likely to produce skewed results, making NCC the more reliable choice.

Part 2

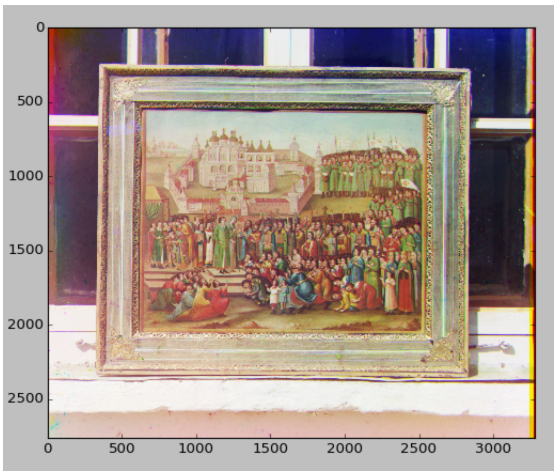
Applying the problem to larger inputs from the dataset.

Earlier, in the description of the dataset, higher-resolution *.png* files were mentioned, but were not dealt with in Part 1 of this assignment. The approach from Part 1 would not produce accurate results for this part of the dataset, due to the discrepancy in size: we would need to test alignments for displacements that are more proportional to the images (10 pixels is not enough). However, making the computations with either method for that amount of displacements would be extremely time-consuming. Therefore, the higher-resolution images will be aligned using an image pyramid, a series that consists of the same image increasing in resolution, coming to its original size at the end (bottom) of the pyramid.

The smaller resolution images in the pyramid will be used to obtain an estimation for the correct displacement, and scores will be taken from the larger images based on the displacement from the smaller images. In my implementation, the pyramid went as small as going to some percentage of the original image width (3%, to be exact), using NCC to incrementally align the images until we got to the full resolution image. At each increase in resolution, we would decrease the range of displacements to check alignments for (because at higher resolutions, checking more displacements can be costly).

The results turned out decent for most images. Not all images were perfectly aligned, but usually only missed by a tiny displacement. Some sample output images (including some comments) can be found in in Figure 6.

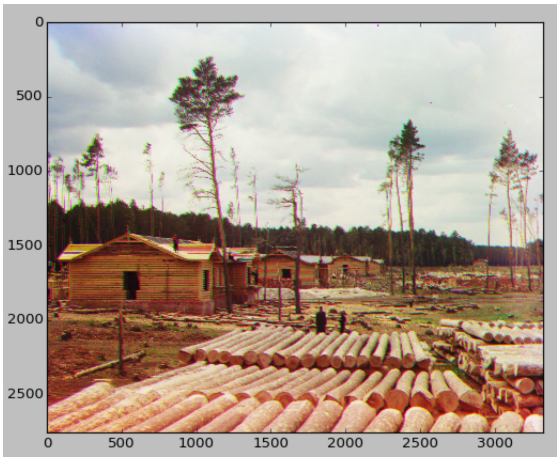
I will estimate an upper-bound for the runtime of this implementation using data from part 1. Considering that running NCC on a lower-resolution file takes about 1 time unit (in my case, my test run was about a second). I predict that because the larger resolution images are about 10 times larger in height and width, I would simply take this time unit and multiply by 100 (first by 10 for height, and then another 10 for width). Though the method for the higher resolution images requires more steps, I believe they balance each other out. For example, though in the lowest level of the pyramid, we are looking at an image that is 10 times as large as the lower resolution images, we are also taking less displacements, which will shave off a bit of time that we can use to find the many displacements for the higher levels on the pyramid. If each time unit is one second, as it was for my own testing, then the runtime for running this method would be about $100 seconds \approx 1.67 minutes$.



(a) Runtime: 1.57219223579 minutes.

(b) Runtime: 1.60740085045 minutes.

(c) Runtime: 1.58106016715 minutes. Note that upon close inspection, the green channel is slightly misaligned.



(d) Runtime: 1.73020026684 minutes. Here, the channels are noticably misaligned. I speculate that maybe there was, at one point, a time where multiple displacements produced the highest score, but my implementation chose one that does not actually align.

Figure 6: High-resolution sample output photos.