CS543/ECE549 Assignment 1

Name: Karan Pandya
NetId: karandp2

Basic alignment:

For the basic alignment method I first applied the SSD method to align the images and it worked decently for most images. When Implementing the SSD approach I stored the SSD values in a matrix and then tried to plot a 2-D map of that matrix which gave me insight into how well the SSD was working since I was able to see the Minimas as the darkest points on the map. Then I found the minimum value of SSD in that matrix and shifted the individual channels accordingly.

The SSD did not work for images having R,G,B channels of very different contrast like the prokudin gorski one.So I tried switching the channels from B,G and B,R to B,G and G,R for that image and got a better result which was apparent from the NCC graph in which I got a sharp change in gradient (a spot) when aligning B,G and G,R instead of a gentle gradient (a blurred spot) when aligning B,G and B,R. In the B,G and G,R case i first aligned the G channel to R and then added the B,G shift to align it to the B channel and it gave a crisp image.

Below are my results of the input image without alignment and the output image with alignment with their displacement vectors.

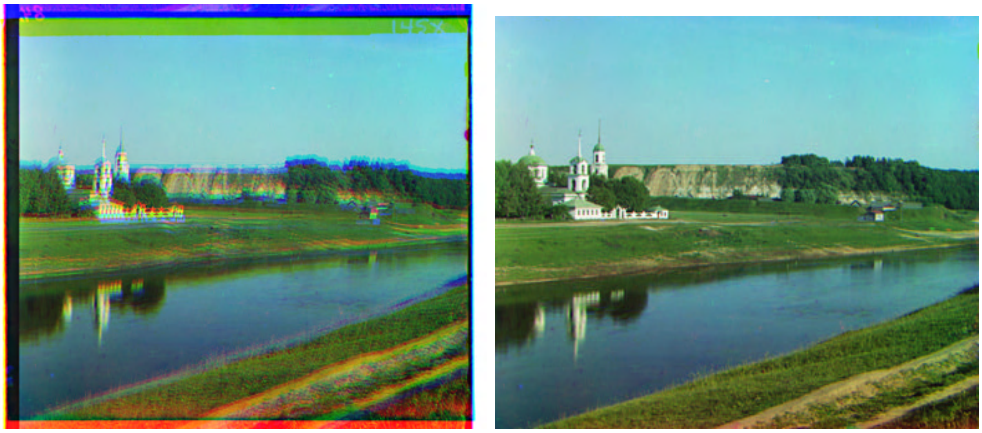
A: Channel Offsets

Using channel <C1> as base channel:

Image	<C2> (h,w) offset	<C3> (h,w) offset
00125v.jpg	[4,2]	[7,1]
00149v.jpg	[4,2]	[4,0]
00153v.jpg	[7,3]	[6,2]
00351v.jpg	[3,0]	[11,1]
00398v.jpg	[4,3]	[9,4]
01112v.jpg	[-1,0]	[3,1]

Without Alignment With Alignment

1)



G Channel displacement vector - [4, 2]
R Channel displacement vector - [7, 1]

2) Without Alignment With Alignment

G Channel displacement vector - [4, 2]
R Channel displacement vector - [4, 0]

3) Without Alignment With Alignment

G Channel displacement vector - [7, 3]
R Channel displacement vector - [6, 2]

4) Without Alignment With Alignment

G Channel displacement vector - [3, 0]
R Channel displacement vector - [11, 1]

5) Without Alignment With Alignment

G Channel displacement vector - [4, 3]
R Channel displacement vector - [9, 4]

6)

Without Alignment With Alignment


G Channel displacement vector - [-1, 0]
R Channel displacement vector - [3, 1]

The SSD algorithm worked faster, nearly double the speed than NCC but did not give the correct results every time when the contrasts of images were very different whereas the NCC algorithm gave accurate and crisp images every time but took more time (double) than SSD.

The limitations of the SSD algorithm was that everytime we had to manually select the best combination of channels to align them in the best way so NCC turned out to be the better algorithm.

Multi-scale Alignment:

For the multi scale alignment I used a scaling factor of 2 to downsample the images. The time taken in the multiscale alignment was significantly less (7x less!) less than the basic alignment solution.

A: Channel Offsets

Using channel <C1> as base channel:

Image	<C2> (h,w) offset	<C3> (h,w) offset
01047u.tif	[22,20]	[66,34]
01657u.tif	[54,8]	[86,8]
01861a.tif	[70,40]	[118,64]


7293

G Channel displacement vector - [54, 8]
R Channel displacement vector - [86, 8]

01657u.tif

Time taken:43.49 seconds

G Channel displacement vector - [70, 40]
R Channel displacement vector - [118, 64]

01861a.tif

Time taken:49.06 seconds

G Channel displacement vector - [22, 20]
R Channel displacement vector - [66, 34]

01047u.tif

Time Taken: 45.96s (By Pyramid alignment method)
Time Taken: 310 seconds (Single scale alignment method)
7x less time!

Bonus Points:

1) Auto_Cropper:

As manual cropping was losing a lot of pixel information I created an auto crop function in which after subtracting the average border size subtracted the maximum of the displacement vector in both x and y direction. This eliminated the red, blue, green bars on the edge of the image caused by the shifting of channels to align them and resulted in a clear crisp image.

Without Autocrop

With Autocrop

As you can see the edge of the river got cut more without autocrop, whereas with auto crop it retained more pixel information.

2) Speeding up the image search.

I was able to reduce the image alignment time for the multi scale images further from 40 seconds to around 10 seconds by searching over a region of image instead of the whole image and using basic alignment search. It gave the same result as pyramid search in 10 seconds. But the main limitation of this method was CHOOSING the RIGHT region for doing the search which was tricky since if it selected a wall/flat region then it didn't work so well but if choosing the region around objects like table, chair, face, it gave a really fast search with just the basic alignment, without using the pyramid method!