CS543/ECE549 Assignment 1

Name: Kaylin Chen
NetId: tc49

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?

For dividing the image, I use the crop method from PIL to separate the image, using a fix 10% ratio to eliminate the white and black borders of each image. I tested with the ratio ranging from 0% to 20% and found that not cropping or over cropping the border will reduce the output alignment.

In the basic alignment section, I normalize the data before finding the offset and utilize a window of [-15,15] SSD method to get the displacement. Something worth mention is that instead of using the concept to slide an image through another one and calculate the overlap part, I try to find which part of the comparing image looks more like the base image. I think it is easier to implement by simply cropping the image, however, by this method I need to add a padding to ensure cropping doesn't create new black border that will lead to large SSD value and the final offset need to multiply by -1 to be the actual displacement amount.

For multiscale alignment, I create a 5-layer image pyramid that is scaled by a factor of two for each layer using PIL resize function. I tried to use more layers but the result get worse and the runtime is longer, I think the possible reason is the image with too low resolution will have smaller area to compute the SSD due to my padding strategy and thus finding a bad offset.

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	(5,2)	(10,1)
00149v.jpg	(4,2)	(9,2)
00153v.jpg	Use different base	Use different base
00351v.jpg	(4,1)	(13,1)
00398v.jpg	(5,3)	(11,4)
01112v.jpg	(0,0)	(5,1)

Using channel G as base channel:

Image	R (h,w) offset	B (h,w) offset
00153v.jpg	(-14,-6)	(-7,-2)

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 R as base channel:

Image	B (h,w) offset	G (h,w) offset
01047u.tif	(-72,-36)	(-48,-16)
01657u.tif	(-118,-16)	(-64, -4)
01861a.tif	(-148,-64)	(-80,-32)

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.

It takes about 20 seconds to run the single-scale solution using a window of [-5, 5], but we need at least a window of [-150, 150] to get the best results according to the displacement of the multiscale solution, which can lead to a runtime of more than 5 hr. On the other hand, the multiscale solution takes about one second to find the displacement of the same image.

Part 4 : Bonus Improvements

Post any extra credit details with outputs here.