CSC320A4 report

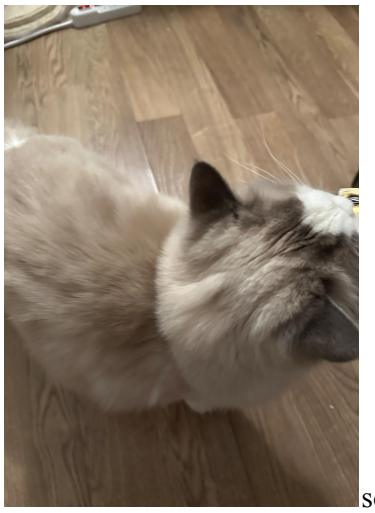
Qiyi Zhang
Student number: 1005723291
Utoid: zhan7393

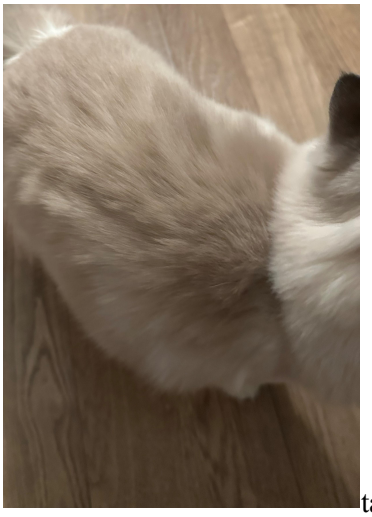
In assignment 4, we explored PatchMatch algorithm. With the help of examples, such as “jagaur.png”, I found it brilliant of the algorithm to reconstruct target from source with the nearest-neighbour field we developed using propagation_and_random_search.

The idea of my propagation_and_random_search function is to firstly initialize best_D with the similarity between each single source patch and its referenced target patch recorded in given bad nearest-neighbour field. Afterwards we do propagation, the field could be improved by checking the similarity between each single source patch and its “shifted” referenced target patch by 1 unit (whether it shifted “up, right” or “down, left” will depend on the odd or even iteration). Lastly, we will do random search to improve nearest-neighbour field . By asserting “w * (alpha ** n) >= 1” and n will plus one each iteration (which guaranteed no infinite loop for alpha = 1/2), we can enter a while loop, each time we will have a offset u , that equals to new_f[y, x] + w * (alpha ** n) * R where R is a random coordinate with x & y between -1 and 1. By checking similarity, again, between each source patch and our new location of target patch, there is the chance for a better match for nearest_neighbour field!

There are few things that I want to address, my algorithm will take about twice to one and a half of the reference time, which is not as expected. I improved it with less calculation in counting similarity and in getting rid of NaN values. However, because I got a lot of errors when using np.geomspace() to eliminate While loop in random search, I finally decided to give it up, which means there is still the space to improve it. Apart from that, I think I implemented the algorithm in the paper well.

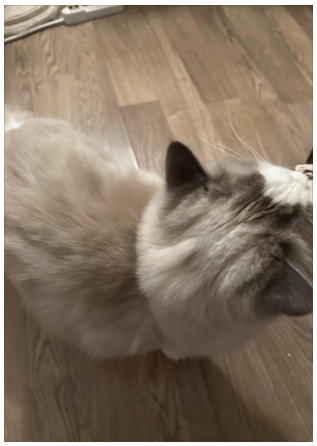
After running through all given examples, I tried algorithm with my examples:
High resolution example:


source

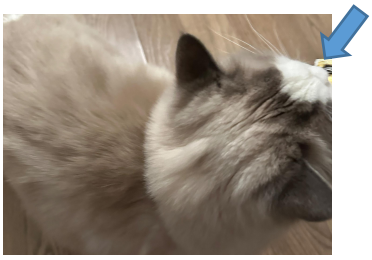

target

After 1 iteration (took about 30 min):

Reconstructed result:



It is very impressive that only using 1 iteration, my algorithm almost perfectly reconstructed the source image, the only difference with the source image is that it lost the color of food that I was feeding my cat (might not be clear):



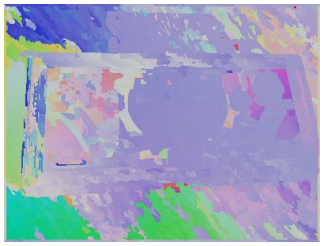
The reason I suspect is that the region for the yellow part of the food is very small and intensity is "outstanding" that it would be hard to find similar path in target patches.

Low resolution example (box of my favorite cookie):

source

target

After 4 iterations:



Reconstructed result:

The low resolution example performs much worse than the high resolution example. As you can see, the generated picture is very blur and it has many artifacts along the pattern of four edges.

The reason I suspect is that since the resolution is very bad, so it is hard to find accurate color path for the patterns on the box. Hence the blur image.

In conclusion, after experiments, I think in this algorithm, unlike in assignment 3, the better resolution image pairs will produce better results. Also, I think the color intensity of the missing part will also affect the result, if the missing part has very small or its color intensity is very different, the result of this part will have artifacts.

Note:
I am grateful for such interesting assignment in such awesome course. Thank you!