CGRA252 Report

Assignment 2

James La

lajame – 300562008

Introduction

In this project, I have created 3 files containing code for core part 1, core part 2, completion. They each use the Patch-Match algorithm to find the correspondences of each patch and certain regions of another image.

I have named each file accordingly and each function in each file according to the procedure performed. The main Patch-Match classes will contain the following functions:

- Initialization function, where it sets the necessary variables.
- Function to initialize the nearest neighbor fields.
- Function to calculate the costs.
- Function to propagate the algorithm forward and/or backward.
- Function to randomly search for a better match for a better match by finding the cost between two patches.
- Run function that iteratively propagates and randomly searches.
- Reconstruct function that reconstructs the final image with the main components found from the above functions.

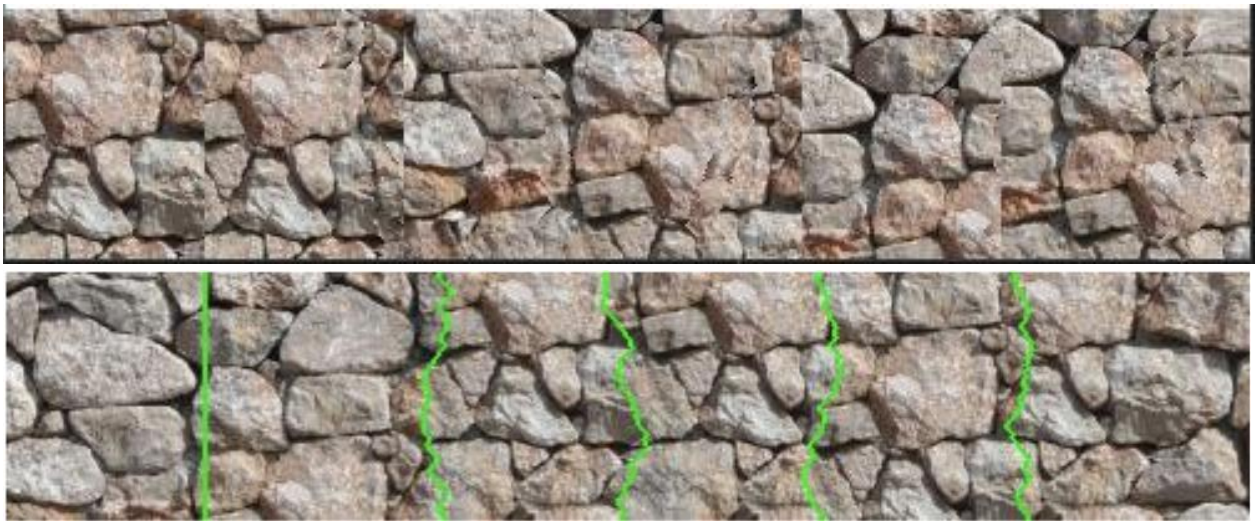
Patch-Match implementation – Core Part 1

This part of the assignment will be in the file “Patchmatch.py”.

As shown in the image above, the resulting image seems to contain some distortedness especially in the more detailed areas. This could be caused by a range of issues. To resolve this, in future Patch match implementations, I could initialize the nearest neighbor fields with a heuristic approach by sampling patches from locations in the source image that are structurally like the target. Additionally, I could adjust the random search so that the radius would not decrease by half at every loop, but at a more optimized rate, so that it would not converge prematurely to the local minima. The last thing I could do is change the error metric so that it is not using the square sum difference but a different error metric like normalized cross-correlation or a gradient-based distance that can handle variations in lighting and noise better.

Image Quilting – Core Part 2

This part of the assignment will be in the file “ImageQuilting.py”.



It seems evident that there are a lot of misalignments in the resulting image. This may be due to a variety of factors, one being the random selection of patches found from the find best patch function that would compare random patches with the current patch. Out of the fifty random patches, if it is unable to find a good patch, the function would still consider one of the fifty to be a good patch even if it isn't.

Another reason this may have occurred is because of the blend width as it may have been too small for certain patches yet, if I had made it bigger, other patches may have been negatively affected. Also, it would have made the synthesize function take longer.

However, there seem to be a few sections of the resulting image that seem to have blended well. This would have been the result of the blend width and a good random patch.

Image Reshuffling – Completion

This part of the assignment will be in the file "ImageReshuffling.py".

Previously, I had a version that would move the ball to the correct location and would fill in the space that originally contained the ball. However, upon realizing that there were minor bugs with the way that it would set the nearest neighbour fields, I modified the code, and it does not work anymore.

In the previous version of the image reshuffling code, although the target patch had changed, it did not contain the ball, but instead contained random pixels. This was most likely the result of my previous bugs.

Image In Painting – Challenge

I have not implemented this part of the assignment.

How to run the programs:

To run each program, you can simply run it through an IDE or code editor.