CGRA352 Assignment 2:

How to run the program:

To run the program, include two arguments, the first argument is the source image for core, and the second argument is the target image for core. In main, the system also has some hard coded images for the completion and challenge. It reads the images from the res folder, having the provided images in the res folder should run smoothly. In the main class, there are booleans to trigger if the core or completion will run or not. The challenge part of the assignment has been implemented to be interactive. Drag and release on screen to change the offset of the ball (or other input). When moving, the algorithm will use less interations to increase speed. When released it will do a full pass.

Patch Match Methods:

1. Initalize():

The Initalize method takes two ints, the number of rows and number of columns. It sets each pixel in the Matrix offsets (which is a field in the NNF class), to a random position within the boundary provided in the parameters. It also sets the Matrix Cost to be Maximum Float.

2. RandomSearch():

First the method initalizes the cost. There is a variable named radius which is either the number of columns or radius depending on which is bigger. For each pixel in the image, it repetitively does the following while radius is still larger than 0. It calculates the Max and min of the radius and column. It then makes sure that the values are within the bounds of the image. Next it randomly gets values between the minimum and maximum rows and columns. Next it calls improveNNF before halving the radius and repeating. Using random search means there is a chance of finding a distant region with a good match.

3. Propagate():

This method looks left to right, top to bottom looking at the previous row and previous column of the current pixel to check if the improveNNF method will improve the offset based on its neighbours. Next this method will do the same but from bottom to top right to left. Following what the original paper does instead of the assignment sheet, this method considers an iteration to be both up and down instead of alternating. It then calls RandomSearch after going up or down.

4.Reconstruct():

The reconstruct method takes a Matrix as a parameter, then for each position of the offsets Maxtrix it it finds the corresponding pixel on the provided Matrix and returns a new Matrix having reconstructed the image.

Image Quilting Methods:

Quilt():

First, this method finds a random 100x100 area as the beginning of the synthesis. It sets the start of the output image to be the 100x100 area. Next we find a patch to overlap the current synthesis on its right by the value of Overlap_Size, using SSD.

We create a new Matrix called overlap and set its values based on the squared difference of pixels in output and texture. From top to bottom we then find the minimum of the 3 pixels above the current pixel and add it to the overlap Matrix. We find the best start position in the bottom row then move up finding the best cuts/seams going from bottom to top. We then create the cut by only adding pixels in the patch to the right of the cut position. This is repeated for 5 iterations. Optionally green pixels can be drawn along the calculated seam.

Reshuffle Methods:

shuffleInitalize():

shuffleInitalize uses the given mask to swap the contents of the mask with the contents of where the mask is moving too. This results in obvious seams where the image sections are shuffled.

gaussianPyramid():

This method creates a pyramid of images, each image is half the size of the previous image. The method pyrDown is used to do this. It is important that pyrDown or some alternative method is used, as this means that pixels are blurred together to reduce loss of detail. The method then patchMatch is called on the lowest pyramid level. Then for a set number of iterations we reconstruct the target using the current NNF and apply patchMatch to improve the ANN. Next, we upsample the offsets. By downsampling and then propagating we are able to optimize the lower levels, using Patchmatch and repeating the process up each level. We are unable to use pyrUp here, as averaging the offsets is not a sensible operation. Instead we double the offset and add 1 if its an odd row or column.

patchReconstruction():

Patch Reconstruction is a better way of reconstructing an image for reshuffling. It uses patches with a set size to recreate the image, taking chunks of images instead of individual pixels. This means that the image will have fewer artifacts after being reconstructed. While the method is not biDirectional, implementing a biDirectional algorithm reduces repetition of ‘objects’ next to each other, by using two ANNs to reduce repetition of areas, with a weighting on patches only repeated once in the image.

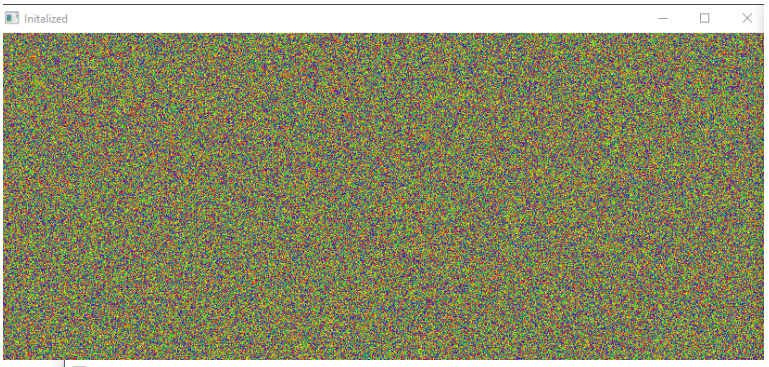
doMouse() and run():

Do mouse and run are a simple implementation to make the Shuffle interactive.

References:

https://link.springer.com/content/pdf/10.1007%2Fs41095-016-0064-2

Images:
Core:

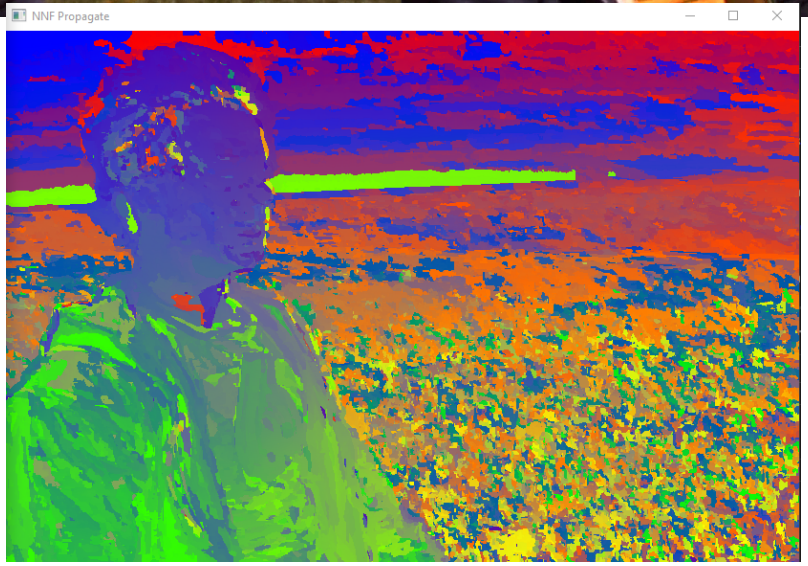

Initalized


Random Search NNF


Random Search


Iteration

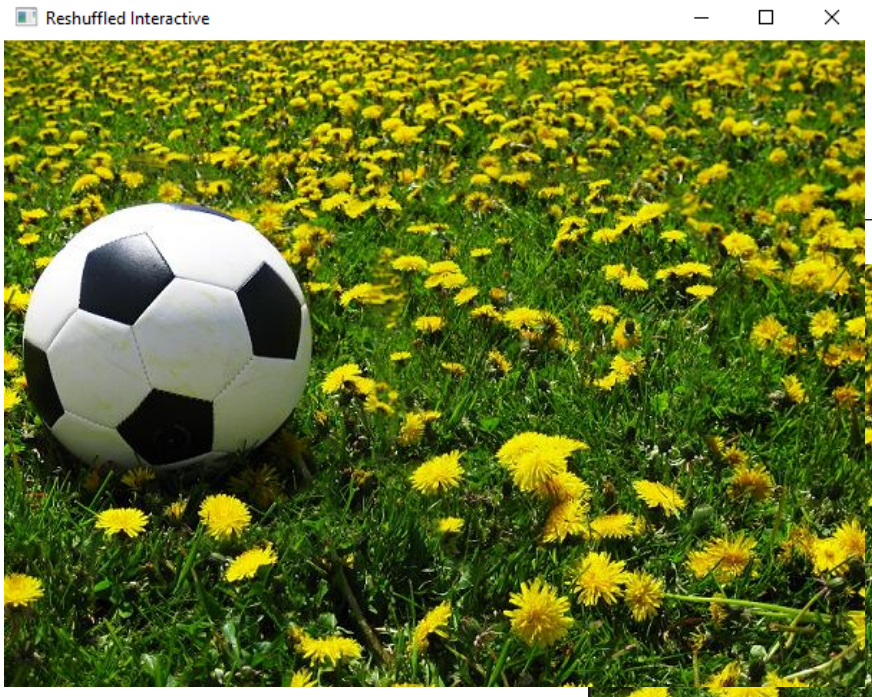

Propagation

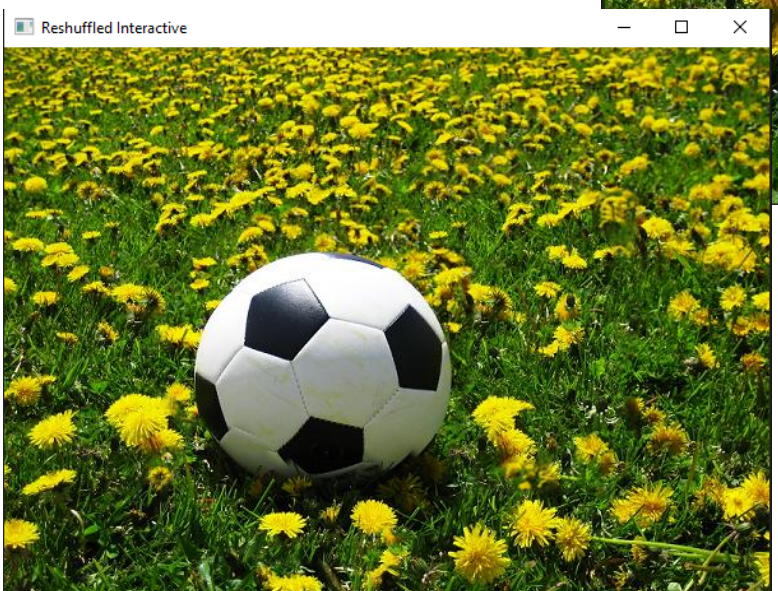

NNF Propagate

Completion:

Note the textures below are different because of the randomization when initalized.

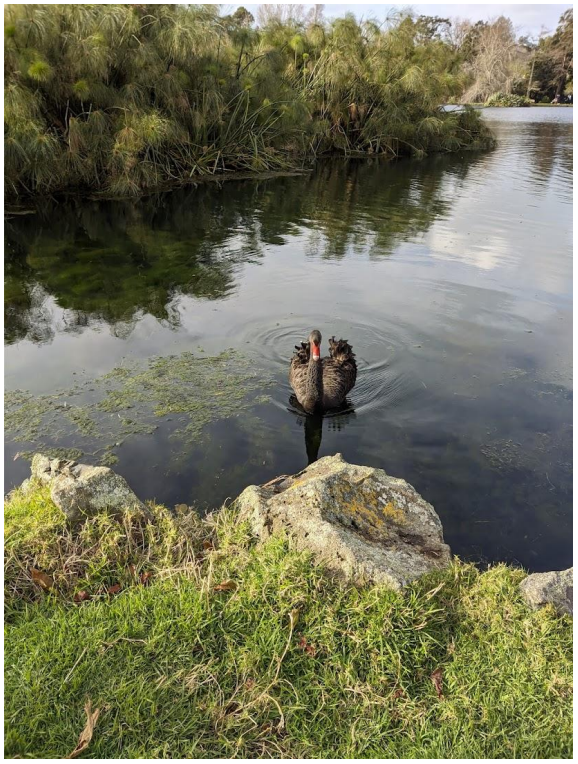
Challenge:





Challenge with my own photo and mask:

Original



After moving