CS283 Assignment 4 - Texture Synthesis and Transfer

Andrew Zhai

For the final project for CS283, I've implemented Texture Synthesis and Transfer. In particular, I've implemented Efros and Freeman's paper on Image Quilting for Texture Synthesis and Transfer.

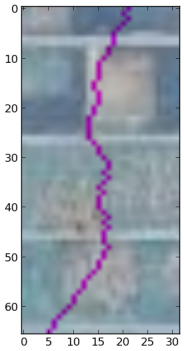
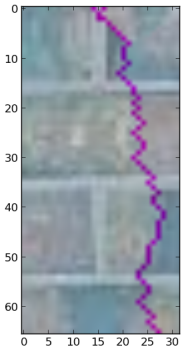
Texture Synthesis

The basic idea of texture synthesis is that given a source texture, you want to use the source texture to create a different texture that has the same pattern as in the source texture. This new texture can differ in size from the source texture. We implemented three different patch algorithms that tries to achieve texture synthesis:

- Random Patches
- Minimum L2 Overlap Patches
- Minimum Cut L2 Overlap Patches

The patch algorithms all have one characteristic in common: they use patches of the source texture to fill the new texture so that the new texture will have a pattern similar to the source texture's pattern. In the **Random Patches** algorithm, we randomly choose patches to fill in the new texture without care of which patch fits best. We care more about how well a patch fits with what's already filled in the new texture with the **Minimum L2 Overlap Patches** algorithm. This algorithm, when choosing a patch to fit in to the new texture, will calculate the L2 error of how well different patches fit in with the neighboring patches and choose the patch that has the least overlap error. The **Minimum Cut L2 Overlap Patches** takes this a step further by optimizing the boundary between the chosen patch and its neighbors through finding the minimum cut between the boundaries.

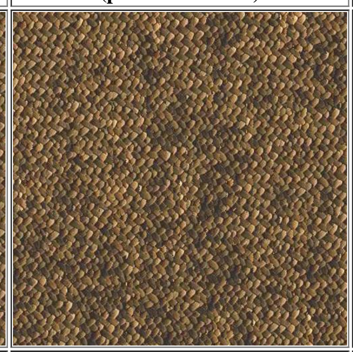
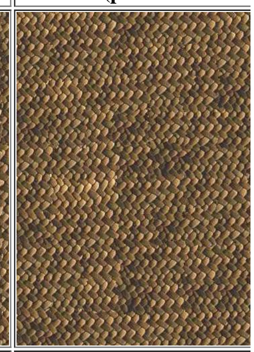
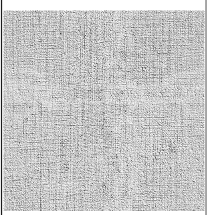
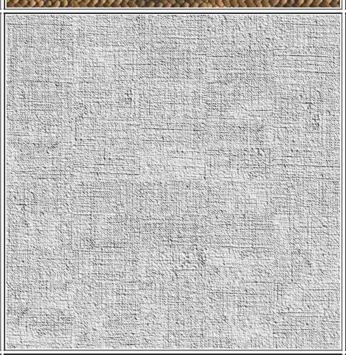
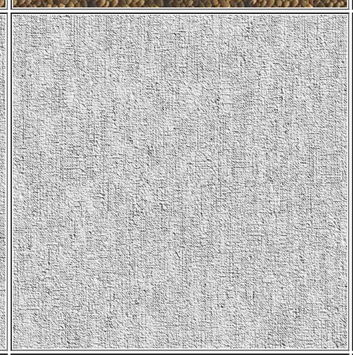
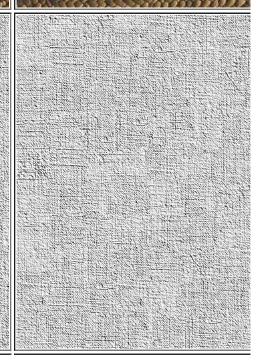
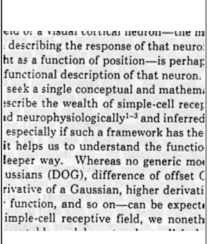
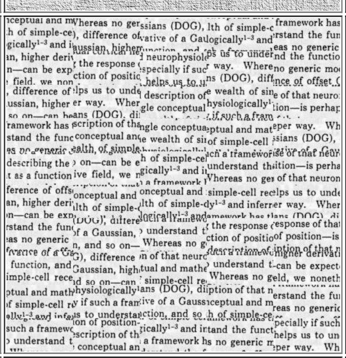
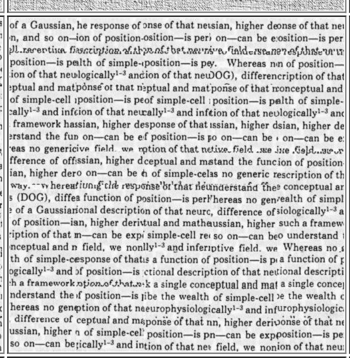
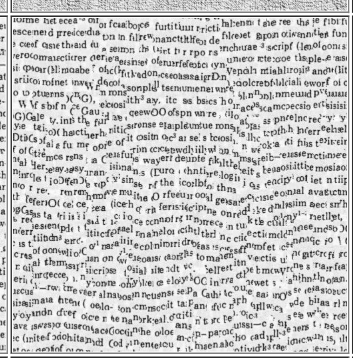
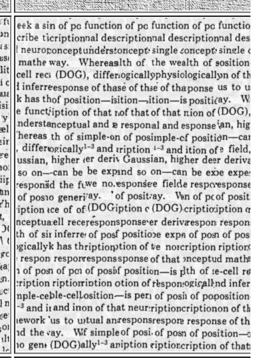
Here are minimum cut examples:

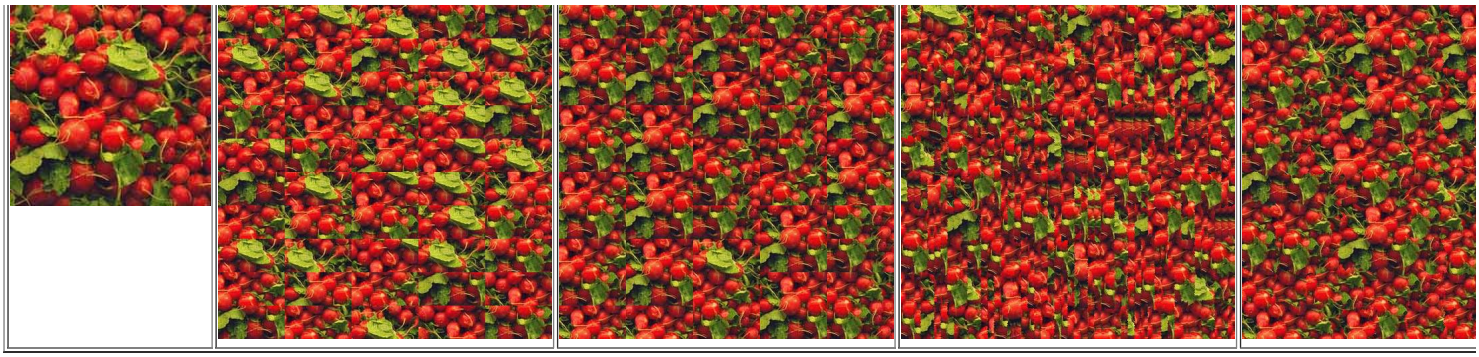
Original Image	Minimum cut
	
	

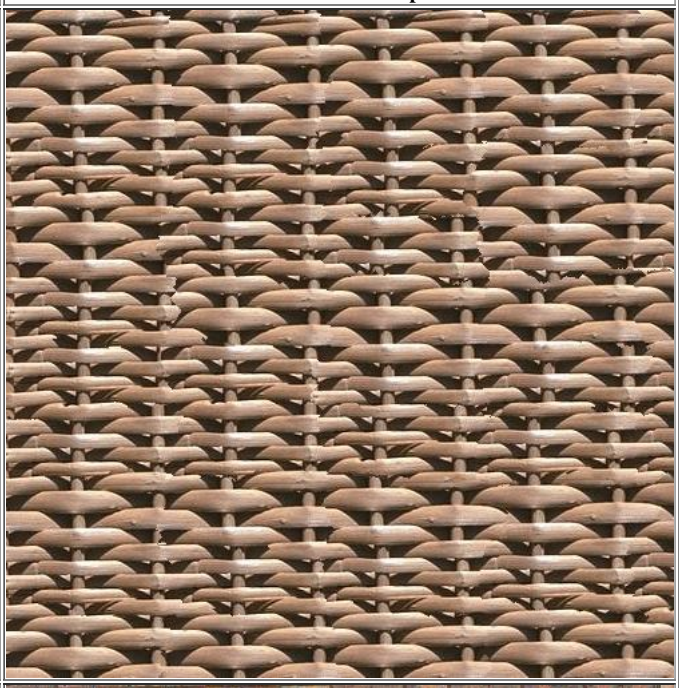
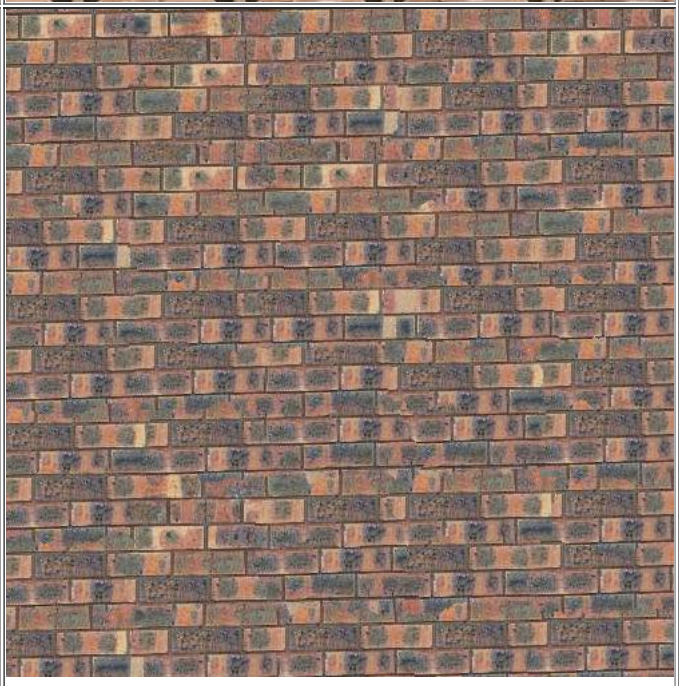
Here is the usage of our texture_synthesizer.py code:

```
usage: texture_synthesizer.py [-h] [-f OUTPUT_FILE] [-pw PATCH_WIDTH]
                              [-ph PATCH_HEIGHT] [-o OVERLAP_PERCENT]
                              type input_file width height

positional arguments:
  type                  Input the synthesizer algorithm to use: 0 - Simple
                        Patches, 1 - Overlap, 2 - Minimum Cut
  input_file            Enter the path to the base texture file
  width                 Input the width of the output texture
  height                Input the height of the output texture

optional arguments:
  -h, --help            show this help message and exit
  -f OUTPUT_FILE, --output_file OUTPUT_FILE
                        The ouput filename for the synthesized texture.
                        Default is 'synthesized_texture.png'
  -pw PATCH_WIDTH, --patch_width PATCH_WIDTH
                        The patch width used. [DEFAULT=50]
  -ph PATCH_HEIGHT, --patch_height PATCH_HEIGHT
                        The patch height used. [DEFAULT=50]
  -o OVERLAP_PERCENT, --overlap_percent OVERLAP_PERCENT
                        The percentage overlap to use. [DEFAULT=.166]. Only
                        affects type 1 and 2
```

The PATCH_WIDTH and PATCH_HEIGHT options control how large the patches will be. The OVERLAP_PERCENT option controls what percentage of the patch will be used when looking at overlap L2 error. To speed up the algorithms that compute overlap L2 error, we optimized by randomly subsampling the exhaustive patch list to create a smaller list of patches so that we don't need to calculate the overlap L2 error for every patch. As a result, we traded quality for speed. This quality drop is however not very noticeable as shown through the examples below:

Source	Random Patches (patch size 100x100)	Minimum L2 Overlap Patches (patch size 100x100)	Minimum Cut L2 Overlap Patches (patch size 10x10)	Minimum Cut L2 Overla (patch size 50x5
				
				
				



Source	Minimum Cut L2 Overlap Patches
	
	

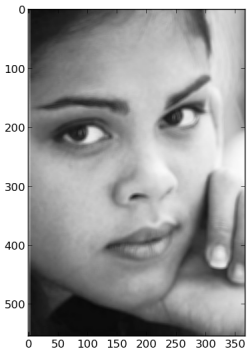
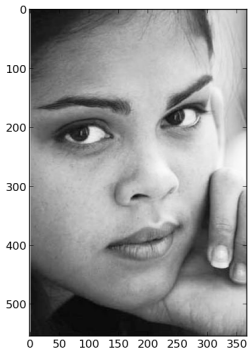
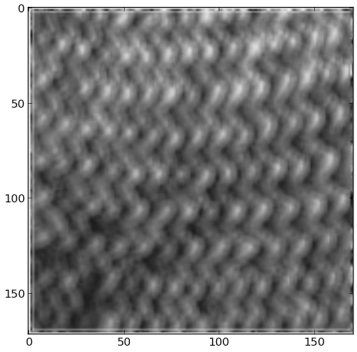
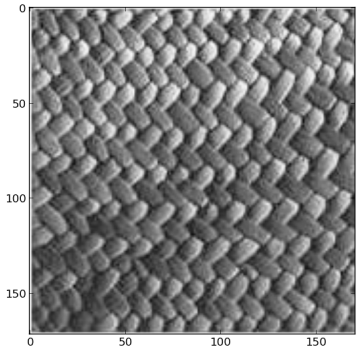
For the above examples, we have fixed the overlap percentage at .1666 as mentioned in the paper.

Texture Transfer

As mentioned in the paper, texture synthesis can be used also for texture transfer by building on top of the Minimum cut algorithm. Instead of having only a source texture, texture transfer also has a target image. The idea of texture transfer is that you are transferring the texture pattern of the source texture onto the target image. This is done as follows: when creating the new texture, instead of only finding patches that fit well with the neighboring patches, you also care about finding a patch that fits well with the target image patch that corresponds to the location in the new image where you're finding the patch for. As a result, you're creating a new image that not only is created from patches from the source texture, but also matches with the target image. Matches with the target image are done through finding the L2 error from the correspondence maps of the source texture and target image. These correspondence maps capture information that wants to be preserved in the new image. We use blurred intensity maps and intensity maps as our correspondence maps.

To improve results, we implemented the iterative version of the texture transfer algorithm mentioned in the paper. During each iteration, instead caring only about the L2 overlap error between a patch and its neighbors and the correspondence map L2 error, we also care now about how well a patch matches what was previously chosen in the new image for the previous iteration. During every consecutive iteration, we reduce the patch size (by an amount specified by user input) and increase alpha based on the formula provided in the paper. This alpha is a measure of the tradeoff between finding the best matching patch to the target image versus finding a patch that fits with the neighboring patches and the overall new image the best. Higher alpha means more care in patches that fits with the neighboring patches and the overall new image the best. As result, we start by finding patches that fit well with the target image and then care more about making the new image fit well with the source texture.

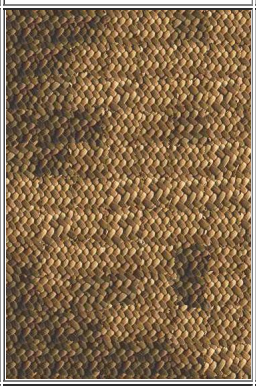
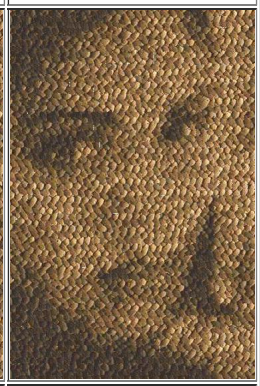
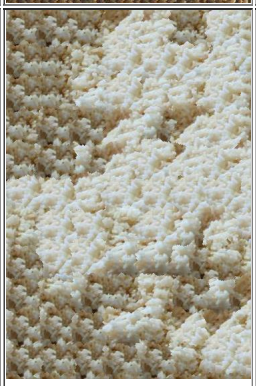
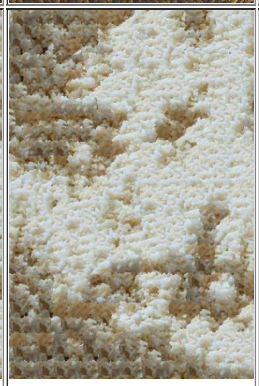
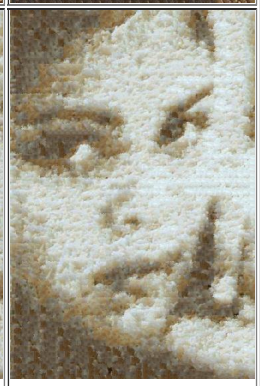
Some example correspondence maps:

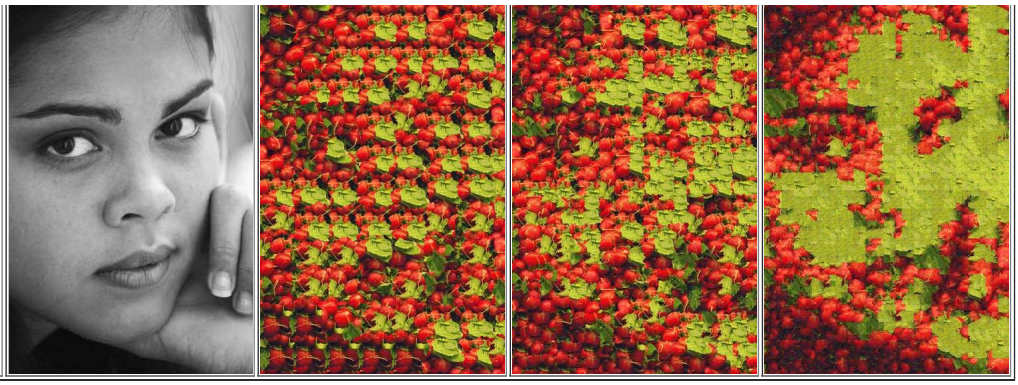
Image	Blur Intensity	Intensity
		
		

Here is the usage of our texture_transfer.py code:

```
usage: texture_transfer.py [-h] [-f OUTPUT_FILE] [-pw PATCH_WIDTH]
                           [-ph PATCH_HEIGHT] [-o OVERLAP_PERCENT]
                           [-r PATCH_LEN_REDUCTION]
                           source_texture target_image corr_map_type
                           iterations

positional arguments:
  source_texture        Enter the path to the source texture file
  target_image          Enter the path to the target image file
  corr_map_type         Input the correspondence map to use: 0 - Intensity, 1
                        - Blurred Intensity
  iterations            Enter the number of iterations to use

optional arguments:
  -h, --help            show this help message and exit
  -f OUTPUT_FILE, --output_file OUTPUT_FILE
                        The ouput filename for the synthesized texture.
                        Default is 'synthesized_texture.png'
  -pw PATCH_WIDTH, --patch_width PATCH_WIDTH
                        The starting patch width used. [DEFAULT=50]
  -ph PATCH_HEIGHT, --patch_height PATCH_HEIGHT
                        The starting patch height used. [DEFAULT=50]
  -o OVERLAP_PERCENT, --overlap_percent OVERLAP_PERCENT
                        The percentage overlap to use. [DEFAULT=.166].
  -r PATCH_LEN_REDUCTION, --patch_len_reduction PATCH_LEN_REDUCTION
                        The percentage decrease in patch length per iteration.
                        [DEFAULT=.333].
```

Here are some results:

Source	Target	Iteration 1	Iteration 3	Iteration 5
				
				
				



Source	Target	Transfer