CSCI 3290 Computational Photography

Assignment 3 – Image Quilting for Texture Synthesis and Transfer

Due Date: **11:59pm on Friday, November 22th, 2013**

1. Backgrounds

The primary goal of this assignment is to implement a classical texture synthesis and transfer method, named **Image Quilting for Texture Synthesis and Transfer** *(SIGGRAPH 2001)* by Alexei A. Efros and William T. Freeman. Texture synthesis is the process of creating an image of arbitrary size from a small sample (grass sample below). Texture Transfer means re-rendering an image in the style of another one (Abraham Lincoln below).

In this project, you are required to implement **the texture synthesis** and **texture transfer** method explained in the paper in MATLAB. You should run both algorithms on at least 3 of your own images in addition to the test cases we provide.

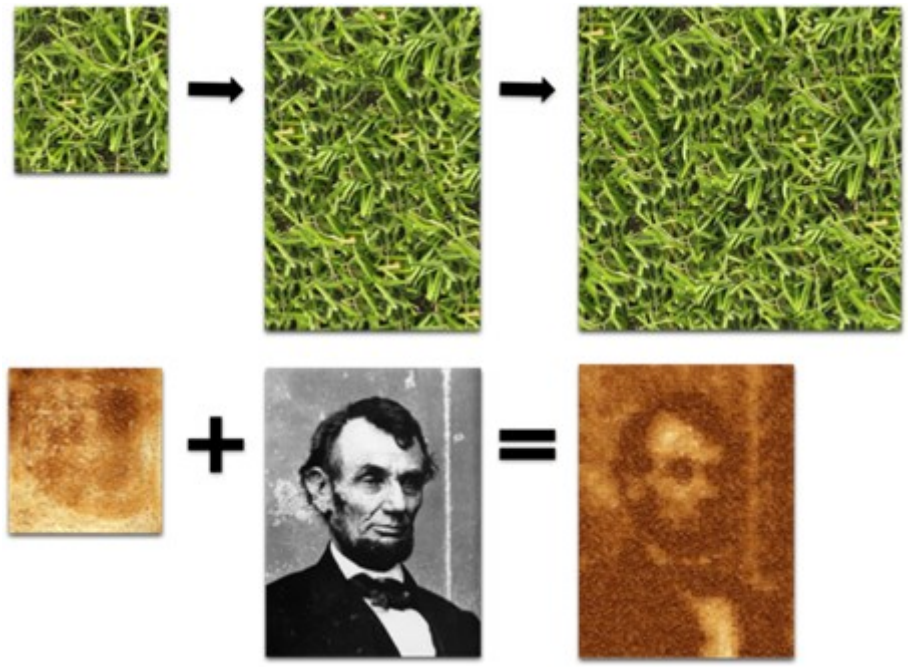

Figure 1. Demonstration of Texture Synthesis and Transfer

Both the texture synthesis and texture transfer are introduced in Chapter 9.

2. Problem

Texture Synthesis

For the texture synthesis algorithm, the general idea is to sample patches from the input texture and compose them in an overlapping way. The simplest solution would be to just randomly select a patch from the input texture each time. With this solution the overlapping regions are not likely to match and this will result in noticeable edges in the result. A better approach is to find a patch in the input texture that has some agreement with the pixels in the overlapping region (e.g. small SSD error). This will already produce pretty good results but still has some unwanted edge artifacts. To get rid of those, your implementation will try to find a minimum error cut for the overlapping region.

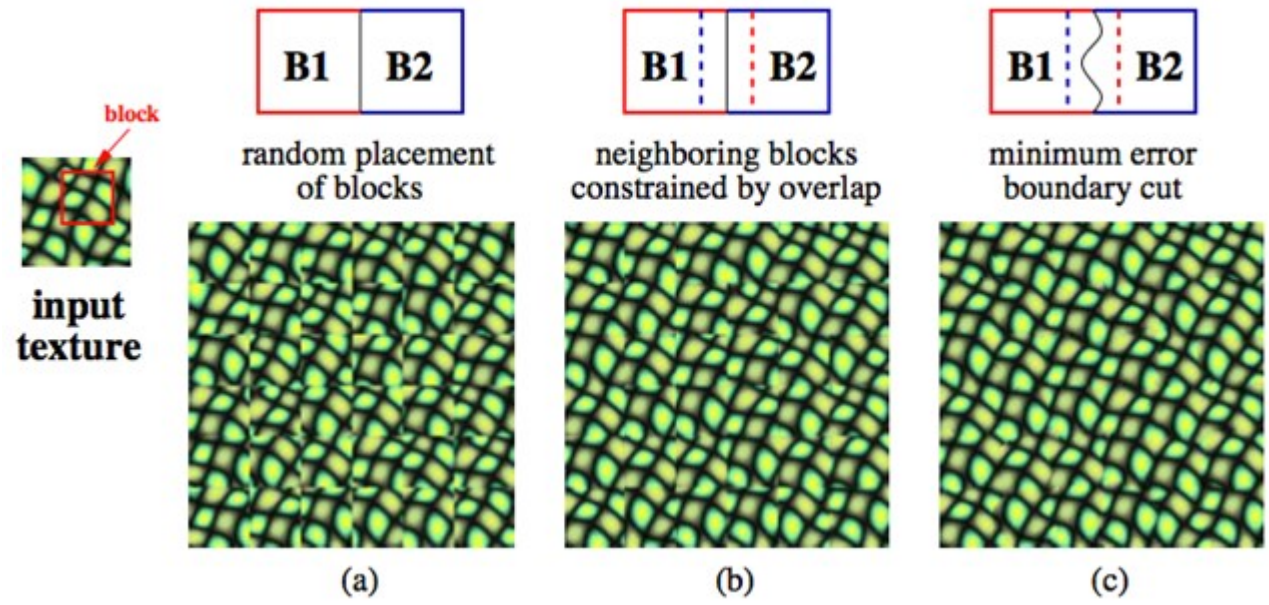

Figure 2. Demonstration of the Three Texture Synthesis Methods

Texture Transfer

We can augment the texture synthesis approach above to get a texture transfer algorithm. That is re-rendering an image with the texture samples of a different image. Each sample patch that we add to our synthesized image must now respect two different constraints: (a) it should have some agreement with the already synthesized parts (this is the constraint we used in texture synthesis), and (b) it should have some correspondence with the image we want re-render. We will use a parameter α to determine the tradeoff between these two constraints. To come up with a term for part (b) we need some measurement of how much a patch agrees with the underlying image. We can do this by calculating the SSD of a patch and the image on some corresponding quantity. One such quantity could be image intensity or the blurred image intensity.

The paper suggest to run multiple iterations of this while decreasing the tile size and adjusting α each time to get the best results. If we run multiple iterations we will need to incorporate the agreement of a patch with

the already synthesized image and not just with the overlap region. So the error term will end up being something like this

$$\text{error} = (\text{overlap\_error} + \text{previous\_synthezised\_error}) + \alpha * \text{correspondence\_error}$$

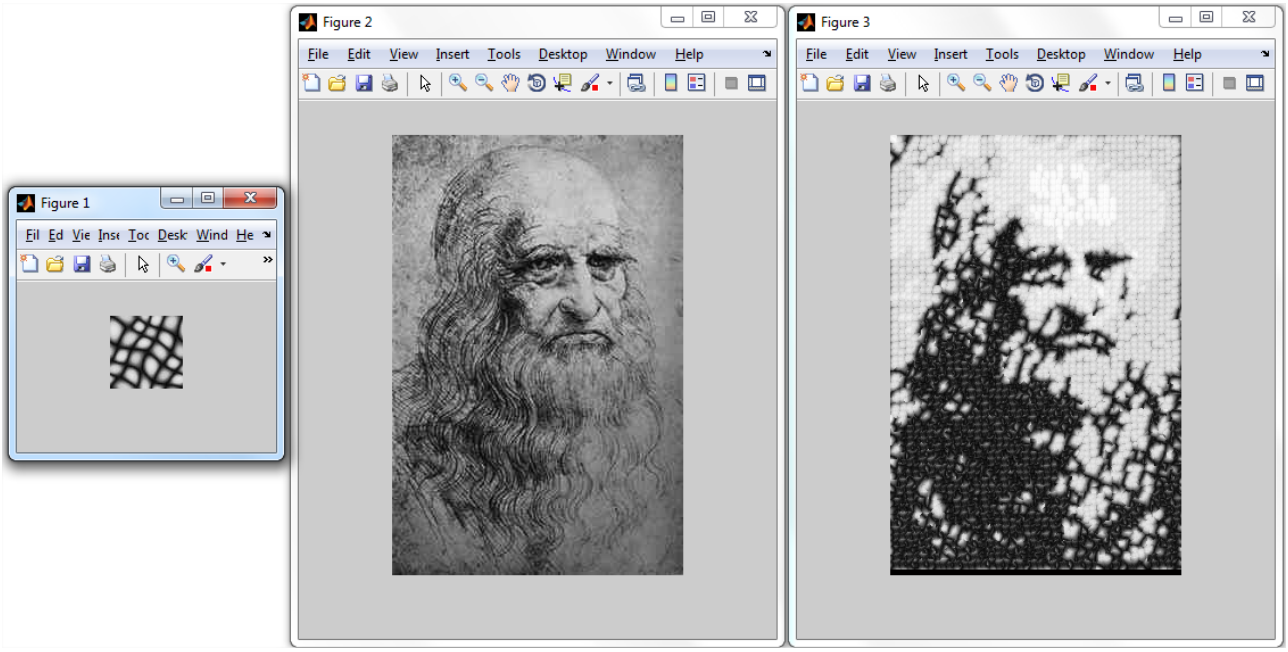

Figure 2. An Example of the Texture Transfer.

Both the two tasks can be finished in one MATLAB function. We can adjust α to change from texture synthesis and texture transfer.

```
function [outputImg] = texture_transfer(inputImg, tarImg, alpha, szPatch, szOverlap, isdebug)
```

3. Algorithms & Implementation Details

The detail of this algorithm is introduced in your lecture notes Chapter 9. To finish your assignment, you can use our skeleton and follow several steps (you can add any function to organize the code in a clear way):

1. Organize your texture completion in four different cases, the initial top-left patch, the top patches of the image, the left patches of the image, and the rest patches.
2. Find out the shared region and target patch.
3. Compute the distance between existing shared texture region and all patches in the input texture image. In this part, you will finish the overlap_error part.

4. Compute the distance between the target patch and all patch in the input texture image for texture transfer. In this part, you will finish the correspondence_error.
5. Find a minimum error boundary and merge the new texture patch to the existing image. (You can use dynamic programming to achieve this goal).

We will give you more implementation hints later.

4. TODO List

We provide you the following files as the skeleton code:

texture_transfer.m, **run_texture_transfer.m**

We also provide some test images:

leonardo.jpg, texture.jpg, yogurt.bmp

More texture examples can be found here

http://www.cc.gatech.edu/cpl/projects/graphcuttextures/

You need to add your own code to complete the skeleton program we provided (you can add any function to organize the code in a clear way):

TODO #1,5,9,13. Find the existing shared region and target patch.

TODO #2,6,10. Compute the distance between existing shared texture region and all patches in the input texture image.

TODO #3,7,11,14. Compute the distance between the target patch and all patches in the input texture image.

TODO #4,8,12. Find a minimum error boundary and merge the new texture patch to the existing image

5. Important Hints

To give you more guidance for finishing your assignment, we provide you some hints about the implementation detail (please read this part carefully):

1. Read the paper if you get stuck. It is pretty easy to understand and reveals a lot of the details.
2. In TODO #2,3,6,7,10,11,14, you can either seek some efficient way to exhaustive search all the patches to computing the distance (avoid using for-loop, and take advantage of existing MATLAB functions), or just random sample a fixed number of patches to speed up the performance.
3. In TODO #2, the error is a bit different from TODO#6, 10 since it contains two overlapping regions.
4. In TODO #4, the minimum error boundary should go across the two regions.
5. Reuse the code for similar functions.

6. Marking

The upper bound score of this assignment is 120.

Basic Part (100%)

(80%) The implementation of the algorithm.

(20%) Report (include results, brief introductions of the two algorithms, your own examples, execution time etc.)

Extra Credit (20%)

The upper bound of extra credit is 20 percent. Be sure to demonstrate your bonus part clearly in your report. You can consider the following ideas:

Bonus 1 (4%): Change your code to support RGB texture synthesis and transfer.

Bonus 2 (6%): If you want to speed things up, you can calculate the SSD between a region and the entire input image (texture) efficiently by using filtering operations.

Bonus 3(4%): The tile size and the overlap size can have a big influence on the result. Run experiments with varying values for those parameters an report your results.

Bonus 4(6%): Experiment with different correspondence quantities for texture transfer and report your findings.

Bonus 5(10%): Try other strategy to find a good cut in the overlapping regions. E.g., Graphcut. Take a look at **Project 3 of CS129's last offering** for some explanation.

7. Submission

Your submission should contain two things: **code files (in MATLAB)** and **a report (in HTML)**

MATLAB Code

1. The completed code files (or your own implementation): **run_texture_transfer.m**, **texture_transfer.m**
2. Any other code you would like to add.

Report Requirements

1. The report must be written in **HTML**.

2. In the report, you should describe your algorithm and any decisions you made to write your algorithm a particular way. You should also show and discuss the results of your algorithm.
3. Discussion on algorithms' efficiency (based on time elapsed).
4. Discuss any extra credit you did.
5. Ideas and algorithms from others **MUST** be clearly claimed. List papers or links in the **Reference** section if needed.
6. Feel free to add any other information you feel is relevant.

Submission Format

The folder you hand in must contain the following:

1. **README.txt** - containing anything about the project that you want to tell the TAs, including brief introduction of the usage of the codes
2. **code/** - directory containing all your code for this assignment.
3. **html/** - directory containing all your html report for this assignment (including images). Your web page should only display compressed images (e.g. jpg or png).
4. **html/index.html** - home page for your results

Please compressed the folder into <*your student ID*>-asgn3**.zip** or <*your student ID*>-asgn3**.rar**, and upload it to e-Learning system.

8. Remarks

1. Five late days total, to be spent wisely.
2. 20% off per day for late submission.
3. Your mark will be deducted if you do not follow the instructions for the submission format.
4. The three assignments are to be completed **INDIVIDUALLY** on your own.
5. You are encouraged to use methods in related academic papers.
6. Absolutely **NO sharing or copying** of code! **NO sharing or copying** of reports! Offender will be given a failure grade and the case will be reported to the faculty.