IMAGE QUILTING

CS 663 - Project

Shrinivas Kulkarni- 180070057

INTRODUCTION

In this project we have implemented the image quilting algorithm presented by Alexei A. Efros and William T. Freeman in their paper "Image Quilting for Texture Synthesis and Transfer".

The algorithm simply takes a portion of texture and is able to generate a larger image, which will be perceived by the human eyes as the same texture. We synthesize new texture by taking patches of existing texture and stitching them in a consistent way.

PROCEDURE

1. Creating a new blank image of desired size

   Here we have defined a magnification factor. So, the size of the final output image will be roughly magnification times the original image's size.

   size(outputTexture) ≅ magnification*size(inputTexture)

   The output image was initialized as zeros of size-

   (round(m*magnification/w)*w,round(n*magnification/w)*w)

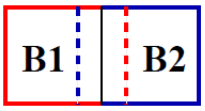
   (m,n) being the size of the original image.

2. Choosing any random patch of a pre-decided size, with extra overlap region

   First of all we will place some random patch, of size (w+o,w+o) taken from the input texture image(we took the right top corner patch).

Now to find the next patch in the first row, we looked for a patch of the same size(w+o,w+o) in the original image. The criterion of choosing the next patch was minimum error in the overlapped region.

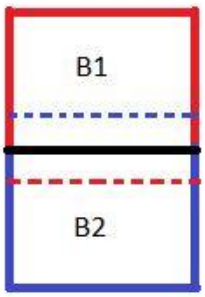
Error meaning the sum of squares of difference between the intensity values in the overlap region from the red and blue patch.



B2 is the new patch added

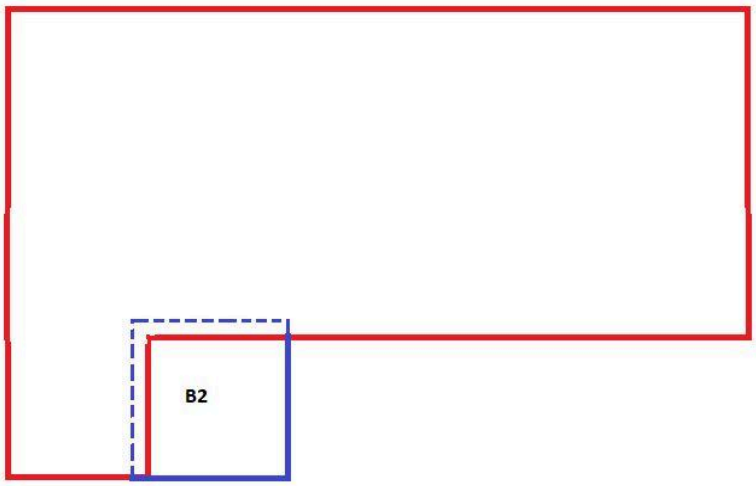
Overlap region is the one between dotted blue and dotted red lines. The dimension of the overlapped region would be (w+o,o). This is only for finding the patches(further which are to be given to the minimum path finder algorithm) in the first row.

To find patches corresponding to the first column, the overlapping region would be horizontal i.e. of shape, (o,w+o).



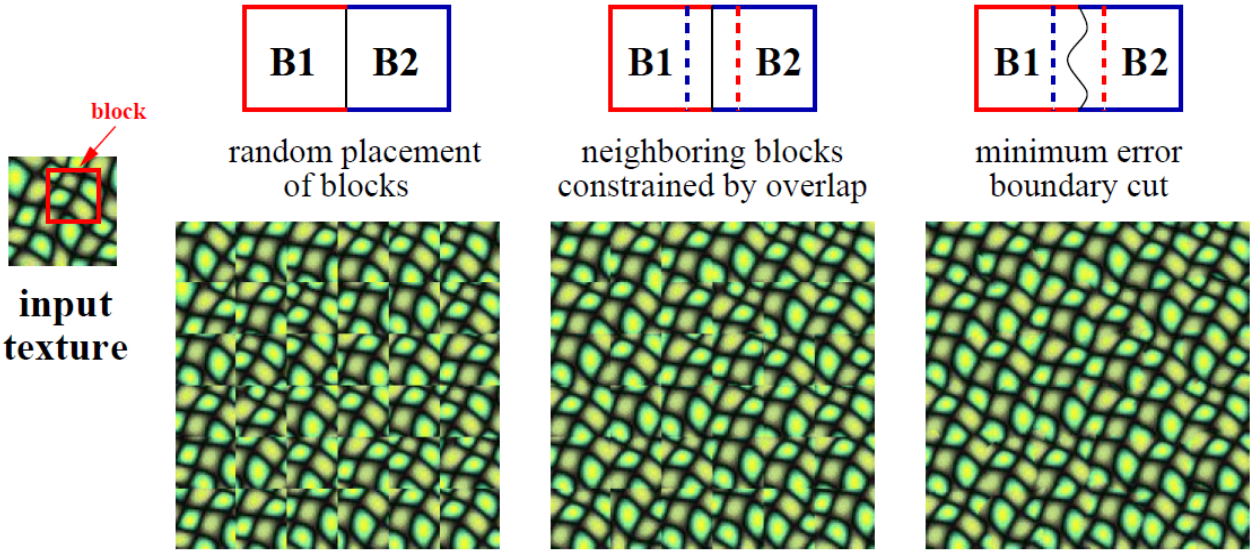
B2 is the new patch added

Now for the case, when the new incoming patch will have a neighbor in left as well as in the top and thus we will be having overlap from top as well as the left patch, as shown below.



So, in this case the overlapping region would be the part of the blue patch in the red region.

The red region is the part of the output image formed till then.



3. When the image is half-created, and we have a new patch to join:
    a. Consider the overlap of new patch and existing image
    b. Define error matrix as the **squared Euclidean distance** between overlaps
    c. Use a Dynamic Programming (DP) approach to find **minimum error path**

d. Use this path as the **boundary** between the patch and image
e. As the path has minimum error over it, we would have a smooth image

The basic formulation of the DP algorithm is,

$$E_{i,j} = e_{i,j} + \min(E_{i-1,j-1}, E_{i-1,j}, E_{i-1,j+1}).$$

We can also use a greedy algorithm (Dijkstra's algorithm) as the matrix can be thought of as a connected graph with edges between pixels within the 8-neighborhood of each other, except the top and bottom pixels.

The methodology was, we used a recursive approach to find the minimum error path, at the same time storing the values already computed in a lookup table to reduce the time complexity of the code.

For any given error matrix, the following pseudocode was used

def method(matrix,column,row):

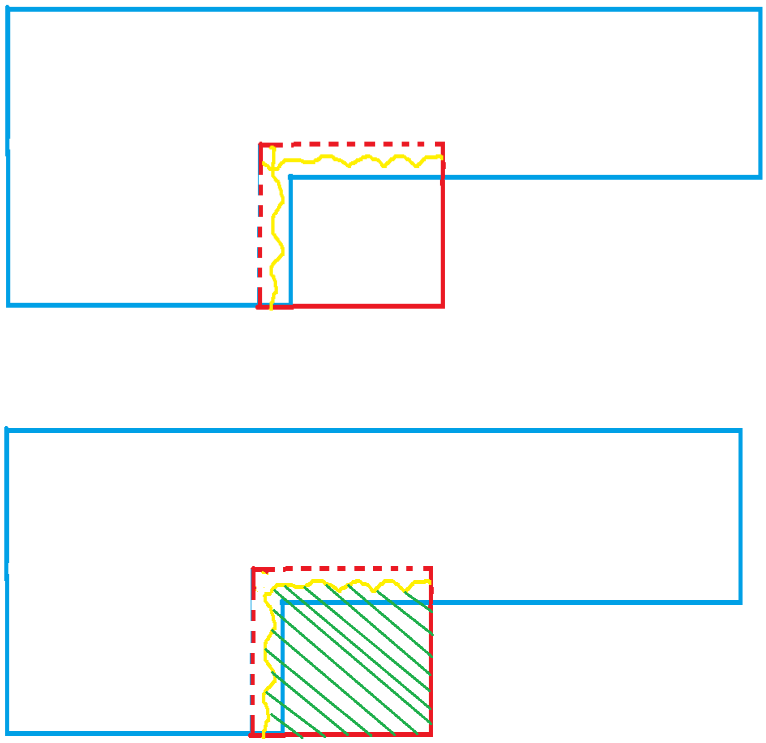
*If (**row,column**) belong to the lookup table:*

***return** lookup(row,column)*

lookup(row,column) := matrix(row,column) + min(next 3 methods)

And then minimising the returned value for the above method for the first Column. Once we have the minimum path length lookup table for all entries corresponding to all entries in the error matrix, we can find. The minimum error path, by going through the matrix once. From the start of the path (already known) and subtracting corresponding error in each iteration to decide the next step in the path. (This was a cumbersome task as implementing recursive codes in MATLAB is difficult, as matrices cannot be passed by reference)

4.

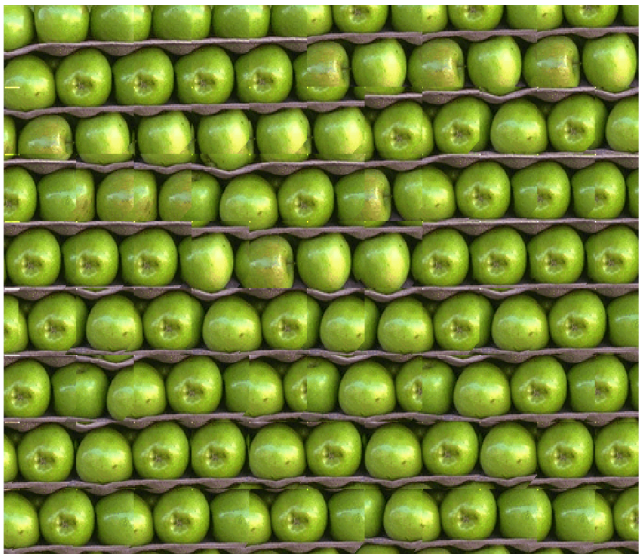


These overlap regions (horizontal and vertical (common region between red and blue)) are given to the dynamic programming algorithm and the minimum error boundary (shown in yellow) is taken from that function. Now, we have two boundaries, one for the horizontal overlap region and the other for the vertical overlap region. These two boundaries will have an intersection in the OxO region in the top left of the red patch. We find the intersection point and then do the following – While constructing the new image, the pixels to the bottom and right of the total boundary are taken from to the new patch and the remaining pixels in the overlap region are taken from the previous constructed image. In the above figure, the green shaded region is from the patch and the rest is from the previously constructed image. In this way, a boundary cut is defined, the new patch is added and the image is formed patch by patch in a raster scan order.

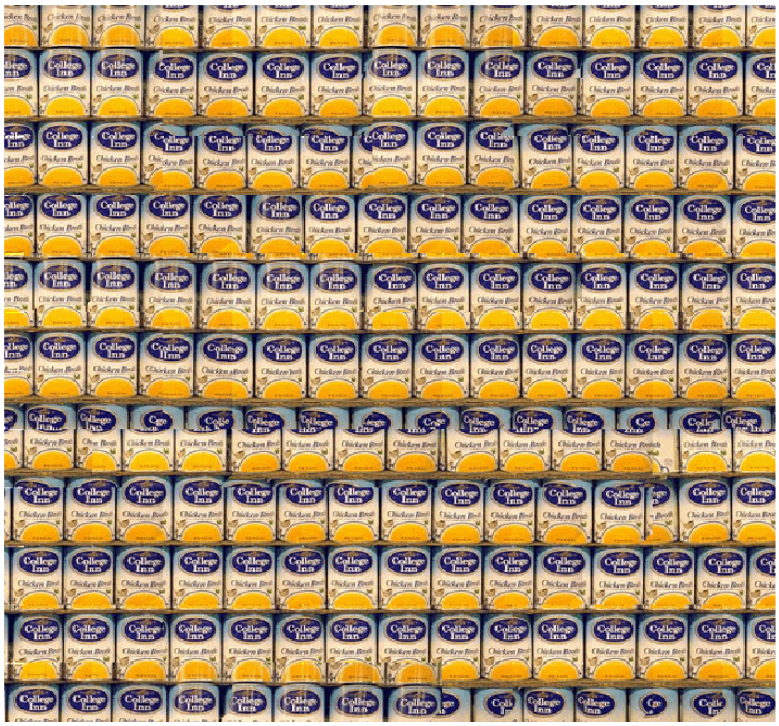
5. Then the constructed image is displayed and compared with the original texture image. Done!

Results -

For all the examples shown, the left image is the texture image and the right image is the generated image.




College Inn
Chicken Broth


College Inn
Chicken Broth