Pranjal Vithlani
Net Id: pv189

Assignment 2

Texture Synthesis

Question 1

Texture Synthesis

Implementing Efros and Leung's approach

A non-parametric method for texture synthesis is proposed. The texture synthesis process grows a new image outward from an initial seed, one pixel at a time. A Markov random field model is assumed, and the conditional distribution of a pixel given all its neighbors synthesized so far is estimated by querying the sample image and finding all similar neighborhoods. The degree of randomness is controlled by a single perceptually intuitive parameter. The method aims at preserving as much local structure as possible and produces good results for a wide variety of synthetic and real-world textures.

Pseudo Code:-

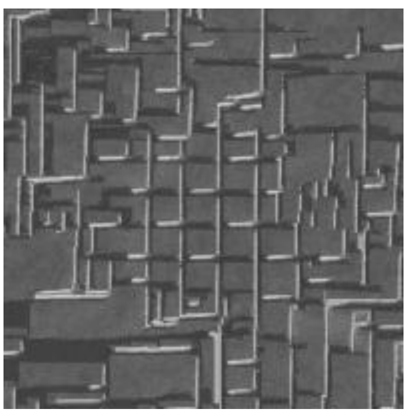
```
function GrowImage(SampleImage,Image,WindowSize)
  while Image not filled do
    progress = 0
    PixelList = GetUnfilledNeighbors(Image)
    foreach Pixel in PixelList do
      Template = GetNeighborhoodWindow(Pixel)
      BestMatches = FindMatches(Template, SampleImage)
      BestMatch = RandomPick(BestMatches)
      if (BestMatch.error < MaxErrThreshold) then
        Pixel.value = BestMatch.value
        progress = 1
      end
    end
    if progress == 0
      then MaxErrThreshold = MaxErrThreshold * 1.1
  end
  return Image
end
```

Pranjal Vithlani
Net Id: pv189

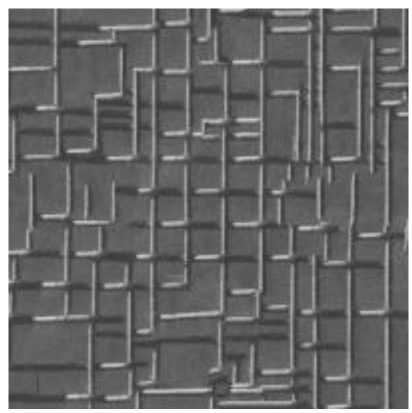
```
function FindMatches(Template,SampleImage)
  ValidMask = 1s where Template is filled, 0s otherwise
  GaussMask = Gaussian2D(WindowSize,Sigma)
  TotWeight = sum i,j GaussiMask(i,j)*ValidMask(i,j)
  for i,j do
    for ii,jj do
      dist = (Template(ii,jj)-SampleImage(i-ii,j-jj))^2
      SSD(i,j) = SSD(i,j) + dist*ValidMask(ii,jj)*GaussMask(ii,jj)
    end
    SSD(i,j) = SSD(i,j) / TotWeight
  end
  PixelList = all pixels (i,j) where SSD(i,j) <= min(SSD)*(1+ErrThreshold)
  return PixelList
end
```

The code for this question can be found in texture__SynthesisQ1.ipynb

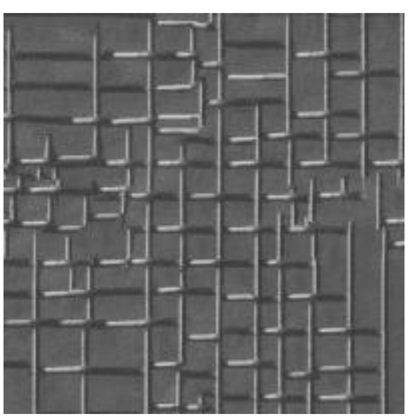
Texture 1



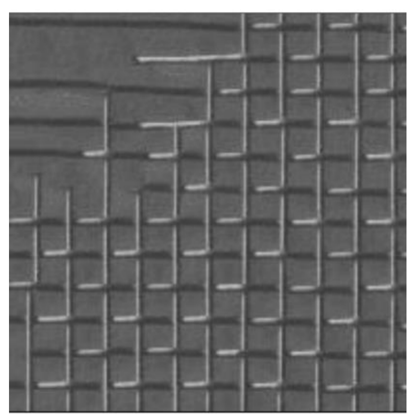
WindowSize = 05



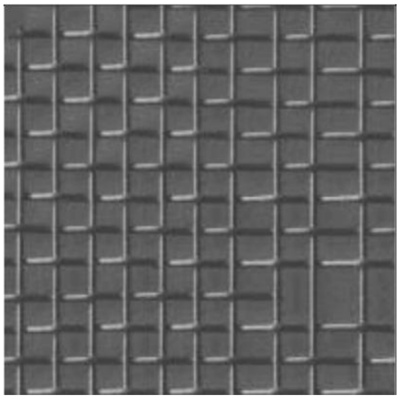
WindowSize = 9



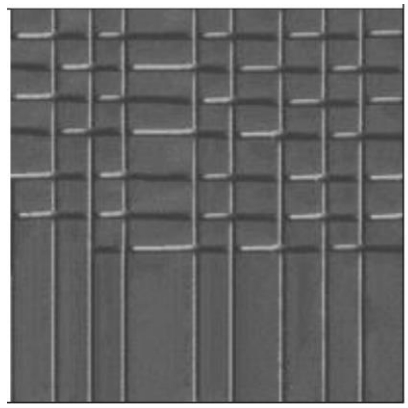
WindowSize = 11



WindowSize = 15

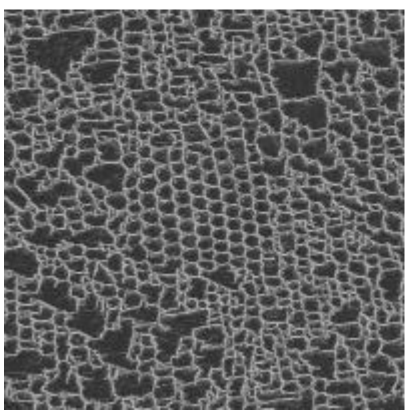


WindowSize = 17

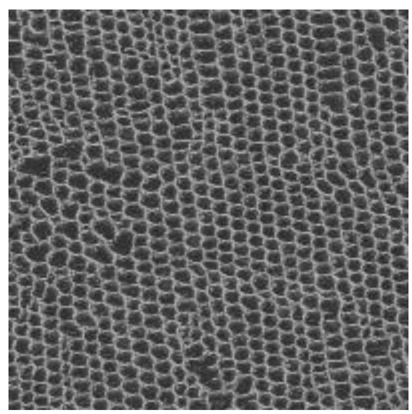


WindowSize = 21

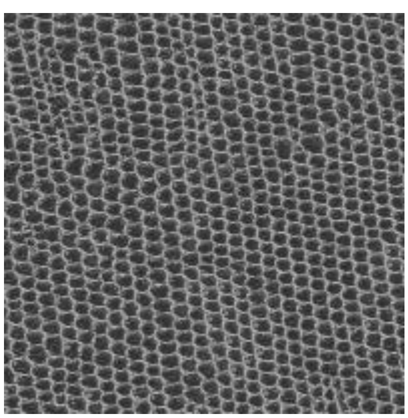
Texture 2



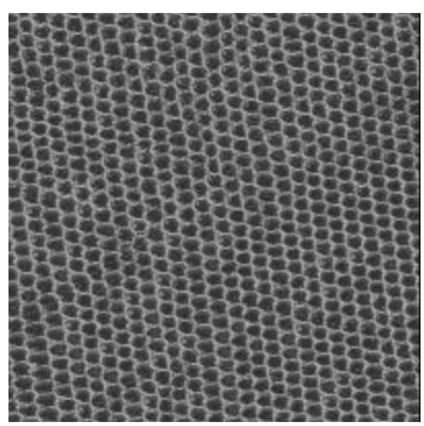
WindowSize = 05



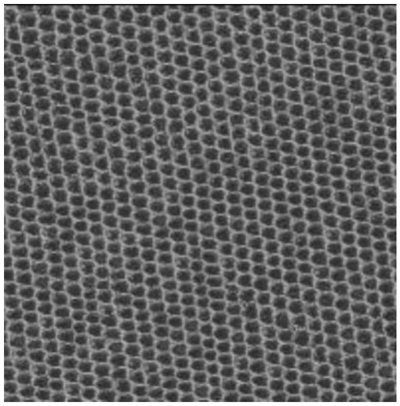
WindowSize = 9



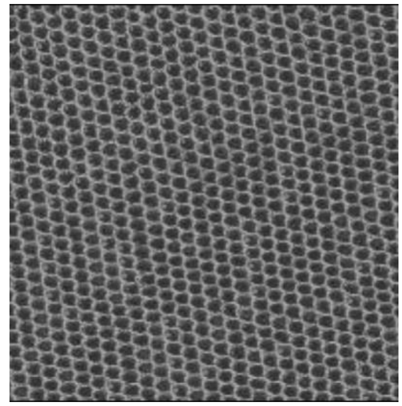
WindowSize = 11



WindowSize = 15

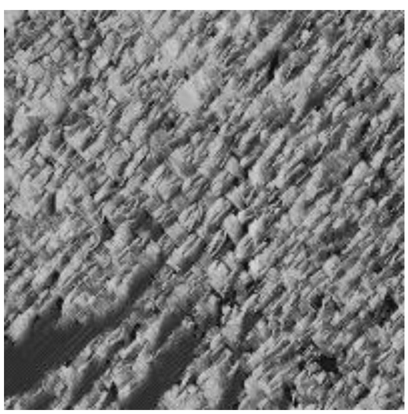


WindowSize = 17

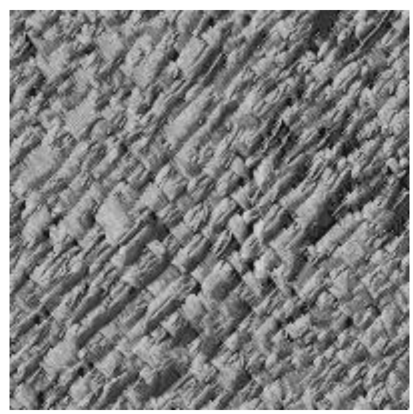


WindowSize = 21

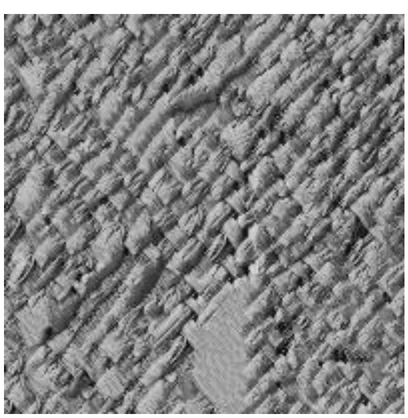
Texture 3



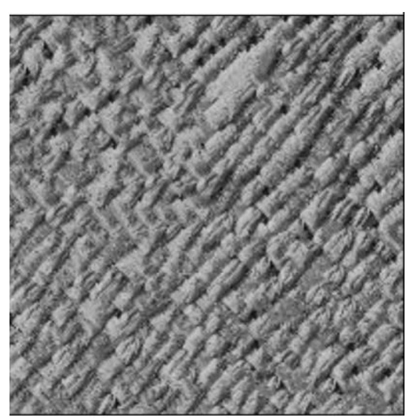
WindowSize = 05



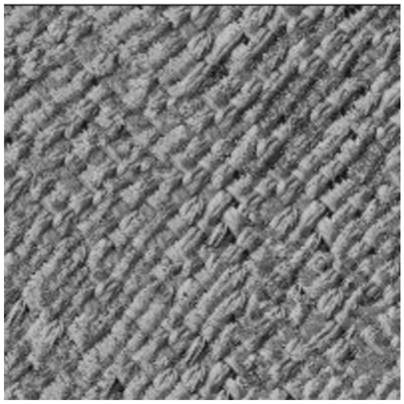
WindowSize = 9



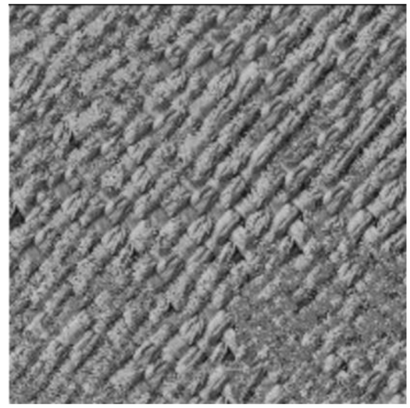
WindowSize = 11



WindowSize = 15

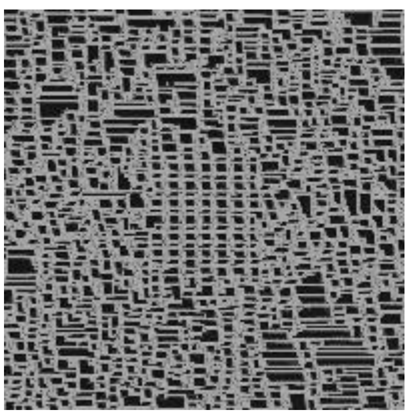


WindowSize = 17

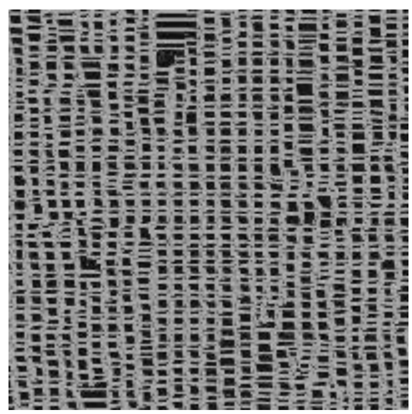


WindowSize = 21

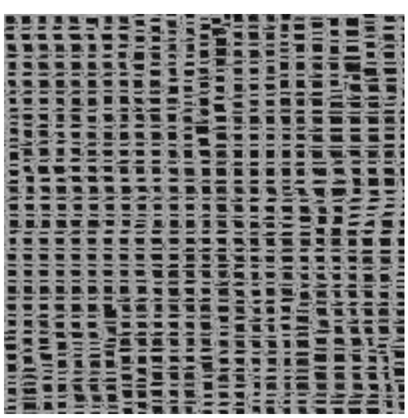
Texture 4



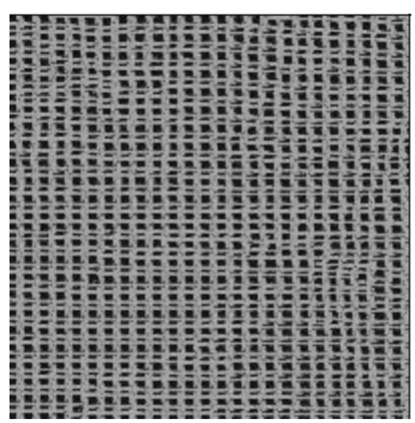
WindowSize = 05



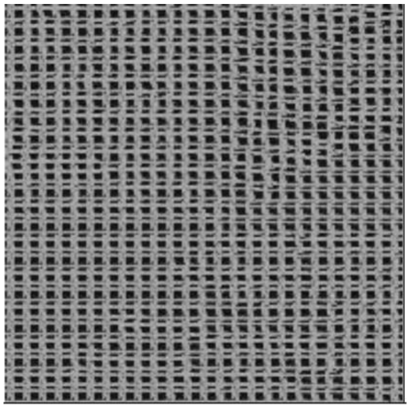
WindowSize = 9



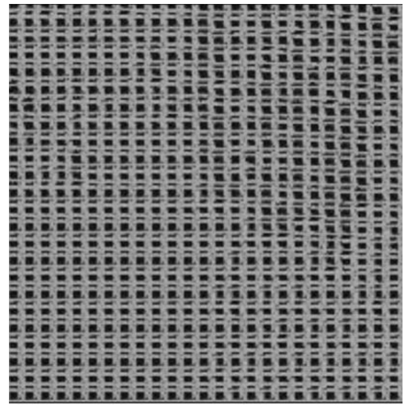
WindowSize = 11



WindowSize = 15

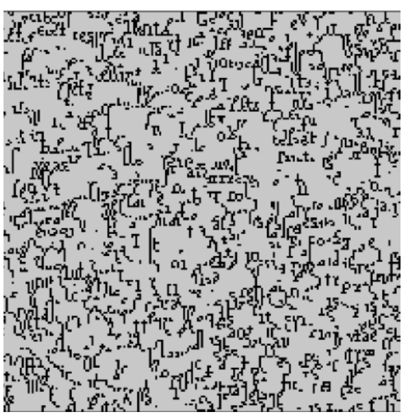


WindowSize = 17

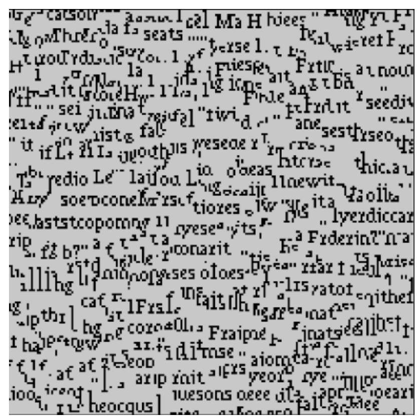


WindowSize = 21

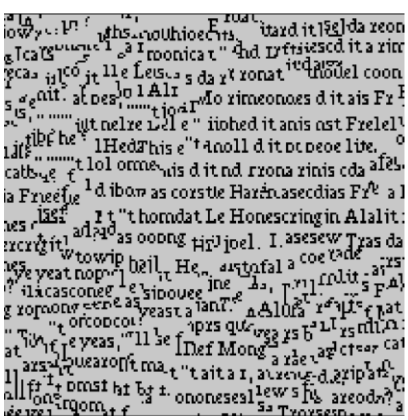
Texture 5



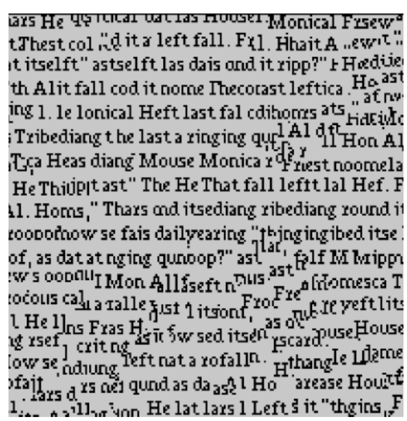
WindowSize = 05



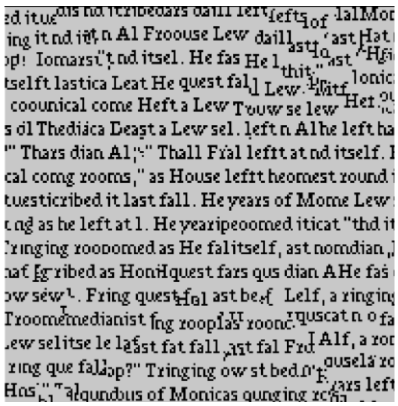
WindowSize = 9



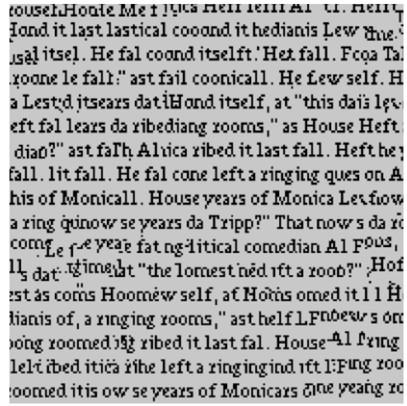
WindowSize = 11



WindowSize = 15



WindowSize = 17



WindowSize = 21

As instructed, WindowSize of 5,9,11 were implemented. But as the results were not satisfactory, I ran the program for WindowSize of 15,17 and 21 also. Results were amazing. First three images started getting better results from WindowSize of 17, while the last two image got better synthesis from the WindowSize of 21. The last two required bigger window to get more information gain from the patch to generate perfect texture while the other three worked fine with the lower patch size as it captured information well and was efficient in run time.

Question 2

Image Inpainting

Using same approach as above. The code for this question can be found in texture__SynthesisQ2.ipynb

Image 1

Original Image

WindowSize = 5

WindowSize = 9

Pranjal Vithlani
Net Id: pv189

WindowSize = 11

WindowSize = 15

WindowSize = 21

Image 2

Original Image

Pranjal Vithlani
Net Id: pv189

WindowSize = 5

WindowSize = 9

WindowSize = 11

WindowSize = 15

Pranjal Vithlani
Net Id: pv189

WindowSize = 21

As you can see if we increase the WindowSize, we'll get better results for Image Inpainting also. WindowSize of 15 and 21 are done just for experiments and I found that performing better.

Question 3

Object Removal

Criminisi, Perez and Toyama, "Region Filling and Object Removal by Exemplar-Based Image Inpainting"

About the algorithm:-

This is ofcourse a better approach, not only in terms of the results, but also in terms of the performance. **This approach directly uses a patch to be filled in the target area, rather than just a pixel in Efros approach.** It selects patch by giving the priority based on 2 features, Confidence and Data. The priority to the patch is given in such a way that it preserves the texture and linear structure. The results were artefact-free! **The order of filling is very critical here.** The pseudo-code from the paper is shown below.

- Extract the manually selected initial front $\delta\Omega^0$.
- Repeat until done:

  1a. Identify the fill front $\delta\Omega^t$. If $\Omega^t = \emptyset$, exit.

  1b. Compute priorities $P(\mathbf{p}) \quad \forall \mathbf{p} \in \delta\Omega^t$.

  2a. Find the patch $\Psi_{\hat{\mathbf{p}}}$ with the maximum priority, *i.e.*, $\hat{\mathbf{p}} = \arg\max_{\mathbf{p} \in \delta\Omega^t} P(\mathbf{p})$

  2b. Find the exemplar $\Psi_{\hat{\mathbf{q}}} \in \Phi$ that minimizes $d(\Psi_{\hat{\mathbf{p}}}, \Psi_{\hat{\mathbf{q}}})$.

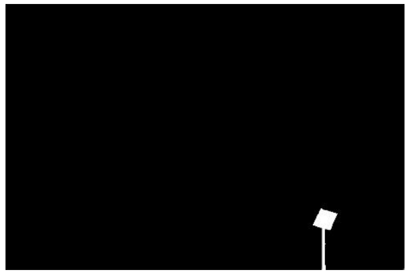
  2c. Copy image data from $\Psi_{\hat{\mathbf{q}}}$ to $\Psi_{\hat{\mathbf{p}}}$ $\forall \mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega$.

  3. Update $C(\mathbf{p}) \;\; \forall \mathbf{p} \in \Psi_{\hat{\mathbf{p}}} \cap \Omega$

Fig - Pseudo Code for Object Removal by Exemplar-Based Image Inpainting

Original Image

Mask_Man



Mask_Sign_Pole

Mask_Path

Results

Man Removed Image

Pranjal Vithlani
Net Id: pv189

Sign Board Removed Image

White Area Removed Image

Now, Comparing the results with Efros and Leung's approach:

Man Removed Image

Sign Board Removed Image

White Area Removed Image

Pranjal Vithlani
Net Id: pv189

Question 4

Image Quilting

Efros and Freeman, “Image Quilting for Texture Synthesis and Transfer”

- Go through the image to be synthesized in raster scan order in steps of one block (minus the overlap).
- For every location, search the input texture for a set of blocks that satisfy the overlap constraints (above and left) within some error tolerance. Randomly pick one such block.
- Compute the error surface between the newly chosen block and the old blocks at the overlap region. Find the minimum cost path along this surface and make that the boundary of the new block. Paste the block onto the texture. Repeat.

Algorithm for Efros and Freeman, “Image Quilting for Texture Synthesis and Transfer”

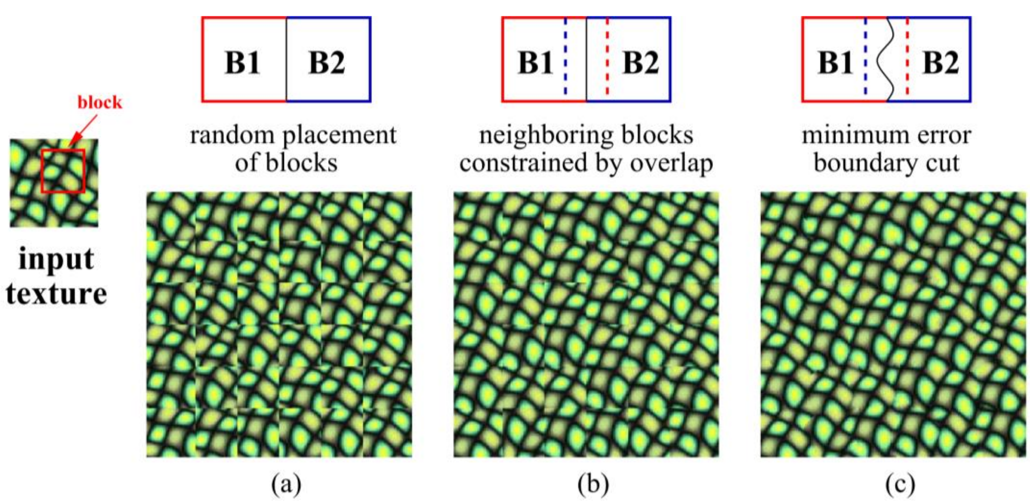
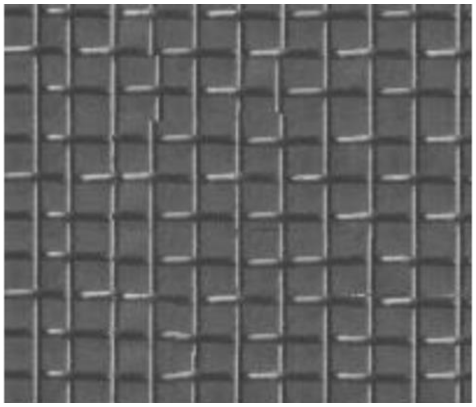


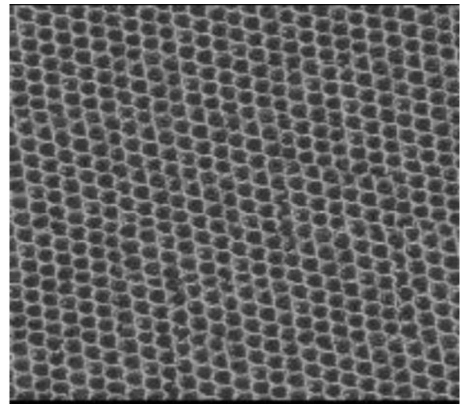
Figure 2: Quilting texture. Square blocks from the input texture are patched together to synthesize a new texture sample: (a) blocks are chosen randomly (similar to [21, 18]), (b) the blocks overlap and each new block is chosen so as to “agree” with its neighbors in the region of overlap, (c) to reduce blockiness the boundary between blocks is computed as a minimum cost path through the error surface at the overlap.

Fig - Approach shown in the visual form

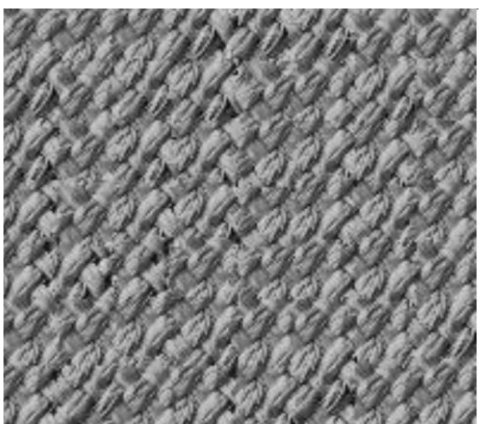
Results



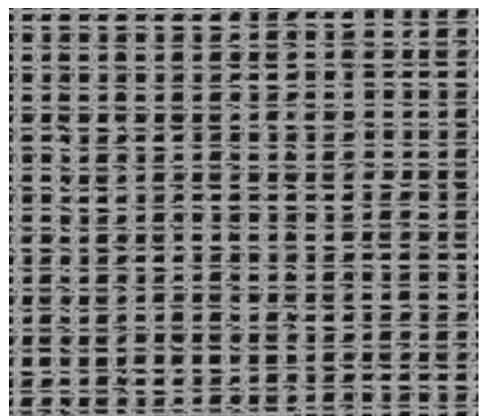
Texture 1



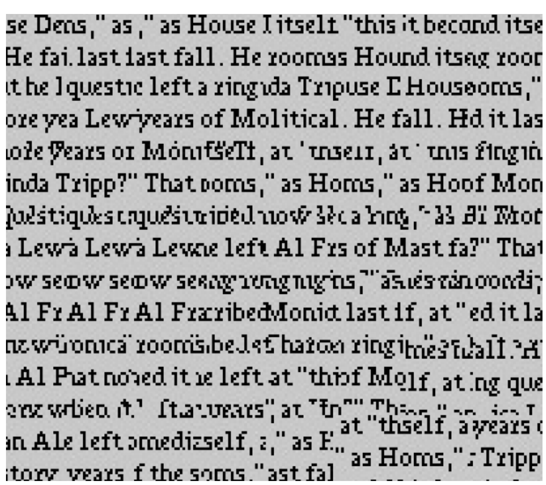
Texture 2



Texture 3



Texture 4



Texture 5

Pranjal Vithlani
Net Id: pv189

Comparison: -

This paper introduces method of synthesizing a new image by stitching together small patches of existing images. Despite its simplicity, this method works remarkably well when applied to texture synthesis, producing results that are equal or better than the Efros & Leung's non-parametric method for texture synthesis but with improved stability (less chance of "growing garbage") and at a very less computational cost. Additionally, method to texture transfer is also provided which gives some very promising results.

To compare running times of best outputs in seconds: -

Texture	1st Method	2nd Method
T1	201	18
T2	154	9
T3	149	8
T4	138	27
T5	570	48

Here, 1st Method means Efros and Leung's approach and 2nd Method means Efros and Freeman's approach

As you can see from the above table the 2nd method takes really less time as compared to the 1st method to generate equally better texture synthesis.